DESIGN AND EVALUATION OF A SYSTEM
FOR THE PRODUCTION OF SHORT DOCUMENTS IN CONTRACTED BRAILLE.

HV1703
.D46c
1975

M.C. MIGEL MEMORIAL LIBRARY
American Foundation for the Blind
15 West 16th Street, New York, New York
10011

C.1

DESIGN AND EVALUATION OF A SYSTEM FOR THE PRODUCTION OF SHORT DOCUMENTS IN CONTRACTED BRAILLE

Warwick Research Unit for the Blind
University of Warwick
Coventry CV4 7AL
England.

1975

HV1703
D460
Copy One

CONTENTS

Introduction

Background

Basic system

Translation program

Evaluation

Conclusions

Acknowledgements

Relevant publications and reports

Appendix 1 - Computer programs

Appendix 2 - Evaluation

Appendix 3 - Further related research projects

Appendix 4 - "Some developments in computer-aided information services for the blind"

Appendix 5 - "Methods of increasing the accessibility of reading materials for the blind"

Appendix 6 - Instructions for typist preparing text for DOTSYS

Introduction

This report covers a one-year period during which a computer-based system for the production of short documents in braille has been initiated. A braille translation program has been implemented and basic problems of input, editing and output have been solved such that an overall system has been satisfactorily constructed. A sufficient quantity of braille has been produced to permit the beginning of a thorough evaluation into the acceptability, type and quantity of demand for this form of braille document.

Background

The University of Warwick started studying aids for the visually handicapped in 1971 when a research study in the discriminability of embossed symbols for points, lines and areas was initiated in close liaison with the Blind Mobility Research Unit at Nottingham University. From this work, the design and evaluation of maps and diagrams evolved, using a unique computer-aided design and production facility. As maps were made available to blind users, the necessity for a considerable amount of accompanying braille text became apparent. Also the recognition of the problem of obtaining single copies of short documents in contracted braille encouraged the University to study computer-based methods for producing braille.

The Sensory Aids Evaluation and Development Center provided Warwick with a copy of DOTSYS III and the Royal National Institute for the Blind financed a one-year project from September 1974. The R.N.I.B. lent the University a Braillemboss and provided funds to employ a programmer, typist/secretary and a part-time braillist. This report summarises the progress attained during this one-year period.

Basic system

The basic system for producing short documents in contracted braille is:

(i) A typist, with no computing knowledge, inputs the text on punched cards, paper tape or directly on a visual display unit. Control characters, for new paragraph etc., are also added by the typist as she inputs the material (i.e. the text is not annotated by someone else).

(ii) A line printer listing of the text is produced in order to proof-read for typing errors.

(iii) The text is interactively edited on a visual display unit with a program designed specifically for this purpose. This program has been designed for speed of operation, minimal computing requirement and for ease of use by operators with no experience of computing.

(iv) The text is translated to a good approximation to Grade II standard English Braille, and the translation is stored on magnetic tape.

(v) The braille is output on an on-line embosser.

Only the translation phase requires extensive central processor time; all other computer operations use less than 1% of the central processor time, and can be time shared with other unrelated programs.

The current output of the system is about 20,000 braille cells (circa 30 pages) per day. Allowing for multiple copies, such that on average about two copies of each document are produced, this means that 15,000 braille cells are translated per day, requiring about one to two minutes of central processor time for translation. It is not envisaged that this can be increased with existing facilities (particularly staff) since the typist is currently the only full time worker on this project, and she also undertakes all proof reading and routine secretarial duties associated with the project.

Translation program

The Sensory Aids Evaluation and Development Center provided Warwick University with a copy of DOTSYS III which is a program, written in Cobol, to translate text to a good approximation to Grade II standard American-English Braille. The version currently in use at Warwick uses 13k words of store with initialisation overlaid and translates at 5000 words per minute to a good approximation to Grade II standard English Braille (see Appendix 1).

Evaluation

In order to evaluate the system a small scale pilot service was operated for producing single copies of short documents for blind subjects. In the first few months of operation the system was undergoing almost continual modification based on informal feedback from the blind subjects which made it impractical to start a formal evaluation programme. However a questionnaire was circulated to subjects who had used the most recent system; the results are shown in Appendix 2.

Firm conclusions cannot be drawn from the survey due to the small sample size. However these results show that both the number of mis-contractions and the use of single-sided braille are acceptable for this application. The lowest score was for turn-round time which varied from a few hours to two weeks for short documents.

Conclusions

This project has demonstrated that there is a considerable demand for short documents in braille, and that computer-based systems can potentially satisfy a significant proportion of this demand.

It is important to continue this evaluation and to carry out the projects listed in Appendix 3. It is recommended that a computer-based service, or services, should be established as soon as possible to meet the unsatisfied demand of the braille-reading population who require short documents in braille.

Acknowledgements

The project is being undertaken with financial assistance from the Royal National Institute for the Blind. The Science Research Council has provided the computing facilities for this project. The Department of Health and Social Security are funding the Senior Research Fellow responsible for the day to day running of the research unit. The aid of these organisations is gratefully acknowledged. We would also like to thank the staff of the Department of Psychology at the University of Warwick who have given considerable help and advice.

Relevant Publications and Reports

Bagley P.R. "A revised system for computer-assisted braille production based on DOTSYS". Information Engineering, USA, April 1973, 15pp.

Coleman P.W.F. "The search for a braille translation program". Computer Weekly, 14 August 1975, p.6.

Coleman P.W.F. "Braille programs - part 2: Defining the language". Computer Weekly, 21st August 1975, p.6.

Douce J.L. "Some developments in computer-aided information services for the blind". Proceedings of the Institution of Electrical Engineers, to be published.

Gerhart, W.R., Millen J.K. & Sullivan J.E. "DOTSYS III: a portable program for grade 2 braille translation". MITRE Corporation, USA, 1971, 139 pp.

Gill J.M. "Methods of increasing the accessibility of reading material by the blind". The Louis Braille Conference, Cambridge, England, Jan 1975, pp 155-162.

Gill J.M. "Non-visual computer peripherals". Research Bulletin of the American Foundation for the Blind, No.29, pp 197-212.

Gill J.M. "The international register of research on blindness and visual impairment". Warwick Research Unit for the Blind, University of Warwick, England, 1975.

Goldish L.H. "Braille in the United States: its production, distribution and use". I.R.I.S., American Foundation for the Blind, New York.

Lawes W.F. "The feasibility study into the usage of computers by and for the blind". Royal National Institute for the Blind, London, Jan 1975.

Mann R.W. "Technology and human rehabilitation: Prostheses for sensory rehabilitation and/or sensory substitution". Advances in Biomedical Engineering, Vol.4, Academic Press, 1974, pp 209-353.

Proceedings of the workshop "Towards the communality of algorithms among braille transcription systems for multilingual usage." University of Munster, Germany, March 1973.

Proceedings of the workshop "Computerised braille production". Danish Association of the Blind, Copenhagen, Sept 1974, to be published.

"A restatement of the lay-out definitions and rules of the standard English braille system". Royal National Institute for the Blind, London, 1969 & 1971.

Appendix 1 Computer programs

Section I

Introduction

This appendix described DOTSYS III-F, a table-driven Fortran program for the translation of English text into a good approximation to grade 2 braille. The program is a translation, with some additions, omissions and modifications, of a Cobol program, DOTSYS III (Gerhart, Millen and Sullivan, 1971). This appendix is based on their report, from which large portions have been quoted with little or no change.

DOTSYS III is a computer program embodying a natural-language-to-braille translation algorithm. The input text is presented to DOTSYS III in the form of 80-character (punched card image) records. These can be manually keypunched directly from inkprint or produced by another program according to a fairly straightforward set of conventions. The output is the sequence of braille signs equivalent to the input text. Output can be produced in either of two forms, "proof" output for a sighted editor and tactile (embossed) braille.

DOTSYS III is almost completely table-driven; i.e., details of the translation algorithm are determined by tables read in at execution time rather than by the program itself. DOTSYS III, as described in this document, comprises both the program and a standard set of tables. In principle, with modified tables, the DOTSYS III program would be capable of processing different kinds of text, such as text containing mathematical or technical notation, languages other than English, or text containing nonstandard symbols for format control.

SECTION II

METHOD

GENERAL

DOTSYS III comprises five cooperating processors: the primary input section, the translator, the stacker, the braille line composer, and the final output writer. It is the translator, as its name implies, that does most of the work of braille translation, and in fact, this section forms a sort of main loop or program, the others being in the form of subprograms called by the translator to do their work at the appropriate time. However, in order to follow the processing of a given piece of text from inkprint form to braille form, it is easiest to imagine the processors running sequentially, each one in turn operating on the output, or a collection of outputs, from the previous processor.

The following descriptions of these processors are idealized to some extent, so that the discussion does not become hopelessly mired in detail.

THE PRIMARY INPUT PROCESSOR

The primary input section reads the 80-characters (card-image) records, The first character of a record logically follows character 80 from the previous record. This stream of characters is collapsed by deleting all instances of the vertical bar character (|), and by deleting all consecutive blanks after the first. This collapsed stream of characters, one at a time, is the output of this section.

THE TRANSLATOR

The Buffer

The translation section operates on the stream of characters issued by primary input. As a first step, these are collected into a ten-character sliding window, called the "buffer", so that a group of characters can be examined.

The Alphabet Table Search

The leftmost character in the buffer is looked up in the alphabet table. In this table, there is exactly one entry for each symbol that may appear in the text, containing, among other things: (a) a code denoting the braille sign for that symbol, (b) the symbol's "input class", and (c) an index to that portion of the contraction table which pertains to this initial letter. An input class is an arbitrary numerical code with three distinct purposes, as will be seen.

The Contraction Table Search

The next step is to search that section of the contraction table indicated for this initial letter. In principle, this search may be visualized as a simple top-to-bottom entry-by-entry sequential search, although in fact a much faster tree-search algorithm (described in detail in Section IV is used. An entry in the contraction table consists of: (a) a string of up to nine characters (ten, counting the implied initial letter); (b) a "right-context class" designator; (c) an input class code; (d) a shift count; and (e) a set of up to four braille sign codes.

For a match to occur on a particular entry, three conditions must be satisfied. First, the entire string for that entry must correspond to the buffer, or left substring thereof. Secondly, the input class for the entry determines a set of "state variable" conditions that must be satisfied. A state variable is a logical switch, having a value "yes" or "no" according to its meaning and the text already translated. For example, a state variable whose meaning is "after a digit" would have the value "yes" if the character immediately preceding the one leftmost in the buffer had been one of the digits 0-9 and would have the value "no" in all other circumstances, including initially. "Meaning" is, strictly speaking, defined completely by entries in another table which governs the setting of these switches, called the transition table. This table will be discussed presently. The mechanism by which the input class selects a set of state variable conditions to be tested is called the decision table; this table is also discussed in more detail in a separate section.

The third condition for a contraction table match to occur is that the right-context class be correct. If the right-context designator for the entry is blank, no right-context condition is imposed. Otherwise, the character immediately to the right of the matched string in the buffer is looked up in the alphabet table to determine its input class. Another table, called the right-context table, is consulted to determine whether that input class is one of those listed as acceptable for this designator - e.g., the designator - "P" (for "punctuation") may apply to input classes 2, 3, 13, and 14.

The contraction table search stops when a match occurs or the end of the table section for that particular initial letter is reached. In the latter event the shift count is taken to be 1, the input class and braille sign code revert to those found for the initial letter, and processing proceeds.

The Buffer Shift

The buffer is then "shifted" left by the amount of the shift count, with new characters from the input stream entering on the right.

Braille Sign Output

The braille sign code(s) are sent to the stacker section, constituting the output of the translator. These may directly represent braille signs, or they may be special control codes as is discussed in the section describing the stacker.

The State Transition

Finally, the input class is used to determine a set of transitions to be applied to the state variables. For each variable, the transition may be specified as "no change", "toggle" (change "yes" to "no" and "no" to "yes"), "set to yes" or "set to no".

After the state variable transition, the process is repeated, beginning with the alphabet table search.

The Decision Table

The decision table determines whether the current settings of the state variables permit the use of a given contraction table entry. It is best represented by a tableau in two parts, as depicted in Figure 1. The upper, or decision portion, has a row for each input class; the lower, or condition portion has one row per state variable. The number of columns is the same for both sections, and this number will depend upon the number of independent tests that may have to be made. The elements of the array are single characters, as given in Figure 1.

Processing of the table for a particular input class consists of a left-to-right scan of the associated row in the upper portion. If a "G" ("Go") symbol is encountered, processing stops with a "yes" decision. If an "F" ("Fail") is encountered, processing is likewise terminated, but with a "no" decision.

If one of the symbols "Y" or "N" is reached, then the corresponding column in the lower portion is compared against the current settings of the state variables. A "Y" in the i^{th} row implies that the i^{th} state variable must have the value "yes"; an "N" demands the value "no" and a dash is satisfied by either setting. If the specified conditions are satisfied for all the state variables, then processing is terminated with a "yes" decision if the upper portion had a "Y" symbol, and "no" if it was an "N". If one or more state variables does not have the value specified, then scanning of the decision section row is resumed.

When a dash is reached in the scan, the scan simply continues with the next column. If the scan encounters the end of the row without otherwise reaching a decision, then the decision becomes "no".

THE STACKER

The stacker operates upon the braille sign codes issued by the translator. In the simplest case, these codes are merely accumulated until the code for the blank braille sign is received - i.e., a braille word is finished. This word, constituting one entry in a stack, is normally then removed from the stack and issued as output to the braille line composer. In exceptional circumstances, two or more words may be held in a stack before release. This may be because the order may have to be changed (in braille, units of measure are placed before the associated numeric quantity), or because the overall length of several words must be computed before issuing the first one - as when centering headings.

All special operations by the stacker, including delayed release and interchange of order, are directed by control codes issued by the translator. These are distinguishable from ordinary braille sign codes in that they have a value greater than 64. (See Appendix 1.4)

The stacker maintains up to three distinct stacks, sharing a common area by borrowing entry fields from a linked free storage list. One stack is used for normal output; a second stack is used to collect the words in a heading (such as a chapter title); the third is used to hold the title (running head) if it is present.

THE BRAILLE LINE COMPOSER

The line composer operates on braille words (stack entries) produced by the stacker. In each case, the length of the word will determine whether it can fit on the current braille line, an internal output buffer. If so, it is simply added thereto. Otherwise, the current line is sent to the final output writer, cleared, and started anew with the current braille word.

The line composer also concerns itself with counting lines to determine a new page condition, page numbering and titling, and similar matters.

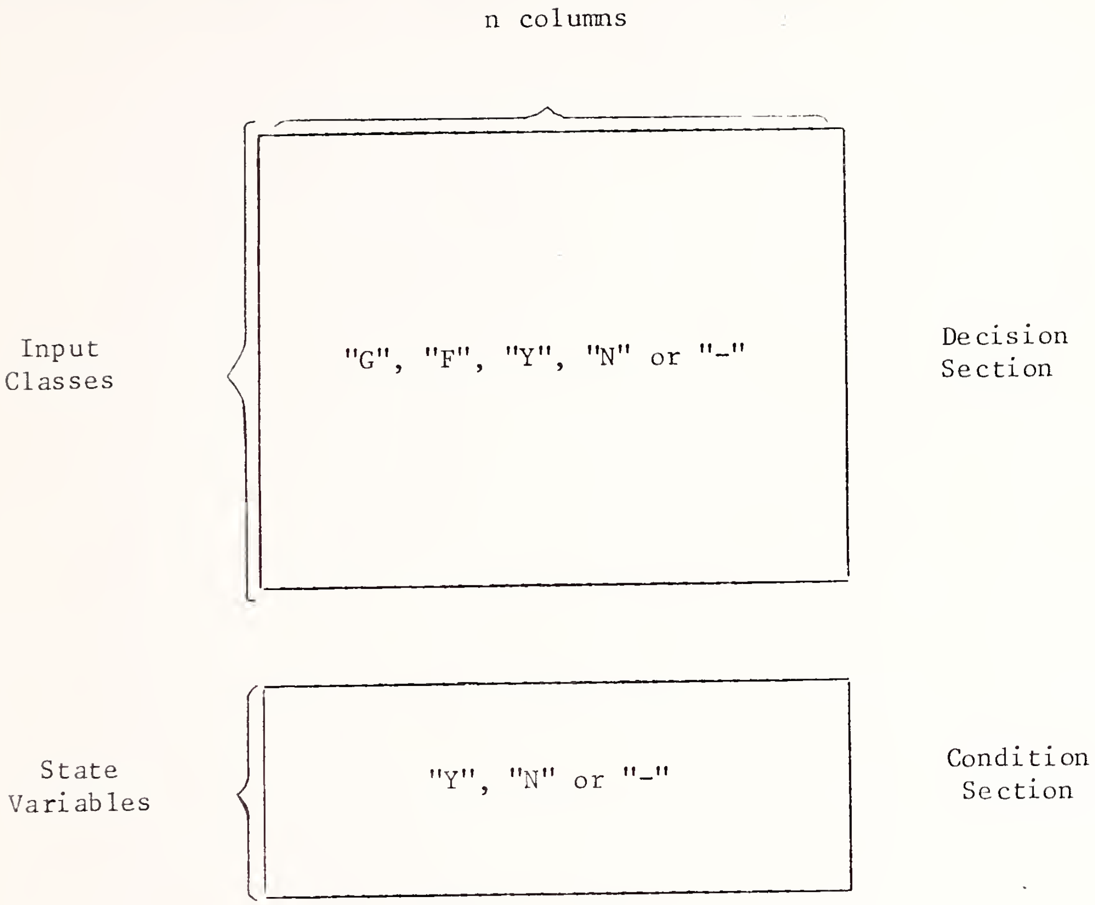


Figure 1. Decision Table Schematic

THE FINAL OUTPUT WRITER

The output writer is actually a collection of processors, one for each distinct mode of output. For each mode selected, the associated processor produces actual output corresponding to the current braille line set up by the line composer.

SECTION III

USAGE

INPUT

Deck Setup

The complete input read by DOTSYS III consists of three main sections, in order:

(1) the tables;
(2) the run control cards; and
(3) the text.

The Tables

Table Data in General

The tables are supplied with the DOTSYS III program and form an integral part of the process described by this document. However, circumstances may arise such that the user may wish to alter or augment the tables. One such circumstance might be a word found to be incorrectly translated by DOTSYS III, that occurs so often in a given text that it is impractical to force the correct translation by special treatment (see the "$/", "/_" and "_/" control symbols under "Forcing and Preventing Contractions" in "Text Input", below). In such a case, an addition to the contraction table is called for.

The tables and associated dimensioning cards are to be arranged in the following order:

(1) the alphabet table;
(2) the contraction table;
(3) the card specifying the number of right-context classes;
(4) the right-context table;
(5) the card specifying the number of state variables;
(6) the card specifying the number of input classes;
(7) the transition table;
(8) the card specifying the number of decision table columns;
(9) the decision table; and
(10) the sign table.

The formats of the tables and cards listed above follow. All two-column numerical fields must contain two digits; in particular, a number less than 10 must be given a leading zero when used in a two-column field.

The Alphabet Table

The alphabet table identifies all legal text input characters. The order of input is arbitrary, but for the sake of efficiency should normally be by decreasing frequency of occurrence of the symbol. (Note also that the order is related to that of the contraction table, q.v.) A dummy entry, with 99 in columns 18-19, should follow the last actual alphabet table entry. Including this card, the total number may not exceed 64. The card format is given in Table I.

Table I

Alphabet Table Card Format

Columns	Contents
1	The symbol being defined.
18-19	The input class (must be in the range from 01 to the number of input classes). Note: A card with 99 in this field signals that the end of the alphabet table has been reached.
21-22	The sign code to use when the symbol occurs in a computer-braille string (see "Text Input", below and Appendix 1.3).
24-25	The sign code to use in normal context (Cf. Appendix 1.3).
30-31	01 if the symbol may be used as one of a pair of symbols bracketing a letter denoting itself; 00 otherwise.

The Contraction Table

The contraction table contains not only contractions but many other sequences related to the heuristics of the translation process, and the definition of special control symbols.

Entries in the contraction table beginning with the same symbol must be grouped together, and these groups must be arranged in the same order as the corresponding symbols in the alphabet table. Also, symbols in the alphabet table that have no corresponding section in the contraction table are grouped, in any order, at the end of the alphabet table.

Within a section of the contraction table dtermined by a common initial letter, all entries having a given second character must also be grouped together. The order of input of these subsections is completely arbitrary and usually will vary from section to section as a function of conditional frequency. This grouping scheme is continued right up to the 10th character. In general, if two entries agree up to the nth character, then all entries between them must also agree with them up to the nth character.

A dummy entry, with 99 in columns 18-19, should follow the last actual entry. The total number of cards, including the dummy, must not exceed 1,000.

Table II gives details of the contraction table card layout.

Table II

Contraction Table Card Format

Columns	Contents
1-10	Character sequence, left-justified. If the string is shorter than 10 characters, then a vertical bar (|) should follow the last character. (Thus blanks are permitted in the string.)
13	Right-context class designation may be specified (non-blank) only if the string is shorter than 10 characters.
18-19	Input class. Note: A card with 99 in this field signifies that the end of the table has been reached.
21-22	Shift count.
24-31	Four two-digit sign codes (Cf. Appendix 1.3). 99's are used for filler on the right.

The Card Specifying the Number of Right Context Classes

As its name implies, this card contains the number of right-context classes that are to be defined. This figure is punched in columns 18-19. It cannot exceed 02.

The Right-Context Table

This table contains one card per right-context class; i.e., there are as many cards as signified on "the Number of Right-Context Classes" card, just previously described. The layout is given in Table III.

Table III

Right-Context Class Card Format

Columns	Contents
13	A single-character designator for the class being defined--e.g., "P" for "punctuation".
24-31	Up to four input class codes. All symbols having one of the listed input classes are implied to have the right context class being defined. If there are fewer than four input class codes, the last one is simply repeated to make four.

The Card Specifying the Number of State Variables

The number of state variables is punched in columns 18-19. This number may not exceed 10.

The Card Specifying the Number of Input Classes

The number of input classes is punched in columns 18-19. This number may not exceed 25.

The Transition Table

This table contains one card per state variable. On each such card, the i^{th} column contains a character signifying how the state variable is to be set when input class i is processed. An "R" signifies "reset" (set to "no"), an "S" means "set" (to "yes"), a dash (-) means "leave", and "T" means "toggle" (change to opposite state).

The Card Specifying the Number of Decision Table Columns

The number of columns in the decision table is punched in columns 18-19. This number cannot exceed 15.

The Decision Table

This table contains one card per decision table column. The layout of each card is given in Table IV.

Table IV

Decision Table Card Format

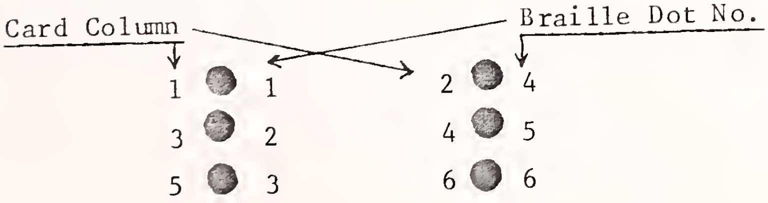
Columns	Contents
1 - nic*	The i^{th} card column contains the upper (decision) section symbol for the input class i. It must be G, F, Y, N or -. (See "The Decision Table".)
(nic + 1)- (nic+nsv)*	The (nic + j)th card column contains the symbol denoting the condition imposed on the jth state variable. It must be Y, N or -.

The Sign Table

The sign table contains exactly 64 cards, one for each "ordinary" braille sign code. (Codes 00 and 64 both refer to sign 64.) The i^{th} card defines code i. The layout is given in Table V.

Table V

Sign Table Card Format

Columns	Contents
1 - 6	Dots, keypunched as periods, corresponding to the braille dots embossed for this character. The correspondence is:  Card Column Braille Dot No. 1 1 3 2 5 3 2 4 4 5 6 6

---

* nic is the number of input classes, nsv the number of state variables.

Columns	Contents
7-9	Up to three characters to be printed on proof output, denoting the most common meaning for this sign.

The Run Control Cards

The run control cards select options that remain in effect throughout the translation of the text-- i.e., for the entire "run" or job step.

There are 5 run control cards. The first two select the output modes (described under "output", below); any combination is permitted. On these, only column 1 is read; it should contain an "N" if the output mode is not wanted or a letter associated with the mode otherwise. Specifically, the options cards are, in order:

(1) Proof output (P or N in column 1).

T in column 16 indicates that a listing of the logic tables is also required.

(2) MIT embosser output (M or N in column 1).

The third and fourth control cards select the braille page dimensions. In both cases the required number is punched in columns 18-19:

(3) The number of braille signs per line (not less than 2 nor greater than 40).

(4) The number of lines per page.

The last card determines whether or not automatic titling and page numbering is to occur:

(5) If the top line of each braille page is to be reserved for a running title and automatically produced page number, a "P" should be punched in column 1 and the starting page number should be punched in columns 32-35. Otherwise, a "N" should be punched in column 1. In this latter case, no automatic page numbering is done and running titles, while still permitted in the input text will not appear on the output.

Text Input

General Keypunching Rules

Text is punched in columns 1 - 80 of each text input card using an EBCDIC keypunch (e.g., IBM 029) or the equivalent multiple punches on another model. Letters, numbers, and punctuation are reproduced as they appear in normal typewritten English text, with the exception of quotation marks, accent marks, mathematical symbols, and brackets ([,]). However, additional symbols must be added to the text to indicate italics and format controls, and to force or prevent improper translations in exceptional cases.

Column 80 of a card is considered to be adjacent to column 1 of the next card. In general, any number of consecutive spaces will be collapsed automatically into a single space (refer to the description of the primary input processor).(e.g.

```
THE START.     THE END.
```

is interpreted as

```
THE START. THE END.)
```

Capitalisation

Where a capital sign is required in braille, a logical-not sign (¬) should be punched. Two logical-not signs (¬¬) should be punched if a double capital sign is required. The capital sign is not normally used.

Italics

If only one, two, or three successive inkprint words are italicized, each of the words must be preceded immediately by an underline (_) when keypunched.

If four or more successive inkprint words are italicized, the first word must be preceded immediately by two underlines (__) and the last by one underline.

Ordering

When two or more special signs must be punched (e.g., an italicised capital letter), the ordering should be:

Italic sign (_)

Capital sign(s) (¬ or ¬ ¬)

Accent sign (¢)

Forcing and Preventing Contractions

The form of any contractable letter group can be forced to occur, regardless of context, by surrounding the letter group with the symbols "/_" and "_/". For example:

a/_dd_/

would cause the "dd" contraction to be used even though otherwise the program would not (and should not) use it at the end of a word.

A contraction that would otherwise occur can be prevented by introducing the null replacement symbol $/ (see below). For example,

DISE$/ASE

will prevent the "EA" contraction.

Replacement Symbols for Special Characters

Mathematical Symbols. The addition sign (+) is represented by $+, the multiplication sign (×) is represented by $X, the subtraction sign (-), is represented by $-, and the division sign (÷) is represented by $DIV. In each case the symbol should be preceded and followed by a space. Other mathematical signs such as > (greater than) and < (less than) must be spelt out as words even though they appear on the keypunch.

Quotation Marks. The single quote character on the keypunch (') is always taken to mean an apostrophe; thus, special symbols must be used for single quotes.

$' followed by a space is punched for a left single quote.

$'R is punched for a right single quote.

Ordinarily, the double quote (") on the keypunch may be used for both left and right double quotes. However, double quotes within the scope of another pair of double quotes must be represented by special symbols.

$" followed by a space is punched for a left double quote within quoted text.

$"R is punched for a right double quote within quoted text.

Accent marks. Any accent or other diacritical mark used with a letter (such as é, è, ê, ä, ç) is represented by preceding the keypunched letter with a cents sign (¢). For example, Abbé is keypunched ABB¢E.

Currency. The pound sign (£) should be represented by a letter L. Where the abbreviation P is used for pence, PENCE should be written out in full.

Brackets.

< is punched for a left bracket ([).

> is punched for a right bracket (]).

Short Syllable Sign. To insert a short syllable sign, the special symbol $SV is used.

Long Syllable Sign. To insert a long syllable sign, the special symbol $LV is used.

End of Poetry Foot Sign. To insert an end of poetry foot sign the special symbol $FT is used.

Caesure Sign. To insert a caesure sign, the special symbol $CS is used.

Null Symbol. The symbol $/ is a null replacement symbol (not a blank) and is generally used to prevent contractions (see above).

Forced Blanks. To produce multiple blanks or otherwise control their placement, the symbol $B is provided. One blank is produced for each occurence of the symbol (e.g., A$B$B$BC produces A   C).

Format and Mode Control Symbols

Keypunching Rule for Spaces Around Format Controls. Unlike replacement symbols where spaces before or after the symbol are interpreted as spaces (i.e., word dividers), any spaces before or after format symbols are ignored. However, the general rule is to provide spaces around each format symbol to avoid possible ambiguity (e.g., $LVOICE would be interpreted as $LV OICE even though $L VOICE may have been intended). Otherwise, the control symbols will usually operate properly regardless of the presence or absence of blanks.

Paragraphs. The symbol $P must be keypunched to indicate the beginning of a new paragraph. The text following the $P is started on the next line after two spaces (that is, starting in the third cell position).

New Line. The symbol $L may be used to begin a new output line. Caution: at most one line will be skipped, even if the $L symbol is repeated. To skip more than one line, see "Skip Multiple Lines".

Skip Multiple Lines. The symbol $SLnnb, where b is a blank and nn is a two digit number (with a leading zero, if necessary) will skip the number of lines indicated, and output will continue from the left margin.

New Page. The symbol $PG may be used to begin a new page.

One-Time Tabulation. If the symbol $TABnnb is punched, where n's represent digits and b represents a blank, the text beginning immediately after the blank will be started at column nn. Tabulation is implemented by the automatic insertion of spaces into the output and will therefore proceed to the next line if nn is less than or equal to the present column number. A leading zero must be supplied if the column number is 9 or less.

Permanent Tabs. Numbered tabs can be set so that the user can right, left, or decimal point justify a word or number on any column. For example, the symbol $STB4L03 means set tab 4 to left justify on column 3. The symbol $STB means to set tab, followed by the one digit number of the tab to be set, followed by an L for left justification

(alignment of the left most character), R for right justification or a D for decimal justification. The last two digit numbers are the column on which justification is to take place. After a tab has been set, it may be executed any number of times by inserting the symbol $# followed by the one digit number of the tab to be executed. (Example: $STB4L03 $#4 LEFT--will left-justify the word LEFT on column 3. The symbol $#4 can be used any number of times.) If a tab is called in a cell position less than or equal to the present position, output will begin on the next line.

The definition of a given tab number may be changed as often as desirable in the text. Initially, all tabs are set to left-justify on column (2 x tab number).

Margin Indentation. The symbol $Mnn will cause the left-hand margin to be set to column nn, so that text will start in column nn+1 of each line. Format control symbols override any margin indentation so that, for example, $L will still cause a new line to start in column 1. Margin indentation may be used to indent continuation lines in an alphabetical list, or for line by line poetry. The symbol $M00 will restore the left-hand margin to the left-hand edge of the page.

Double Spacing. The symbol $DOUBLE will cause subsequent output to appear with a blank line between each line of output. A further $DOUBLE will cause subsequent output to resume single-spacing.

Titles. Titles are placed between the symbols $TLS (Title START) and $TLE (Title END), (e.g., $TLS THIS IS A SAMPLE TITLE $TLE.) After a title has been inserted, each subsequent page has a centred title as its first line (the same line which contains the page number). Pagination must be turned on for titles to appear on output (see "The Run Control Cards"). If a new title is entered, it takes the place of the old on all future pages. Entering the title does not automatically turn to a new page.

Headings. These are similar to titles except that the control symbols are $HDS and $HDE and that headings are a one time occurrence. Headings are centred on the next line with subsequent input beginning in column 1 of the line after the heading. Headings that overflow begin in Column 4 of successive lines.

Poetry. Two alternative forms of poetry input may be used.

(i) If line by line poetry is required, that is, each line of poetry starts a new line of braille: the poetry text should be preceded by the symbol $M04 and followed by the symbol $M00, the first line of each verse should be preceded by the symbol $P and each other line by $L.

(ii) If continuous poetry is required: the poetry text should be preceded by the symbols $P $PY, each verse except the first should be preceded by $P, and at the end of each line except the last line of the poem the symbol $PS should be added without any preceding space.

Computer Braille. An occurrence of the computer braille switch, $C, changes the mode of translation from grade 2 to computer braille or vice versa. It must not be used during grade 1 translation.

Processing several independent documents together. The symbol $T; will reset page numbering to 1, reset state-variables to their initial values, and cancel margin-indentation, double spacing, computer braille, and running title if any of these are in effect.

If several documents are to be processed together it is recommended that the following be inserted at the beginning of the text for each document:

```
$PG $%$%$%$% $%$%$%$% $SL01 <identification>

$SL01 $%$%$%$% $%$%$%$% $T $PG
```

where <identification> is a short description or name for the document. This will produce a distinctive header page for each document in the output, containing the braille translation of the <identification> surrounded by two rows of braille FOR signs (all six dots).

Notes to the Braille Editor

Role of the Editor

This section indicates where the intervention of the editor is required and presents the tools available to the editor to implement his intervention.

There are basically two ways in which the editor can ensure proper translation in problem situations: (1) instructing the keypunch operator to add certain symbols to the input text; and (2) modifying the tables which control DOTSYS III. Modification of tables is explained under "Tables" above; the only time it would be done under normal circumstances would be when a problem word occurs often in a particular body of text; in that case, its correct translation would be added to the contraction table.

In the remainder of this section we shall discuss the special symbols which can be inserted in the input text and the situations in which they are helpful or necessary. These symbols are the letter sign (=), the number sign (#), the grade switch ($G), the division symbol ($/) and the forced-contraction symbols (/_, _/).

Letter Sign (=)

The editor must insert the letter sign when:

(1) a letter which means a letter stands alone and is not followed by a period indicating an abbreviation and is not italicized or surrounded by double quotes or parentheses.

(2) the capitalised or uncapitalised letter "a", "i", or "o" requires a letter sign in the braille, except when used after a number or as a word.

(3) combinations of letters that could be confused with short-form words appear in the input text. (It is suggested that a contraction table entry be used to insert the letter sign if such a letter combination occurs frequently. The tables already contain a number of frequently used abbreviations.)

(4) Roman numerals appear in the text.

(5) abbreviations are used without periods.

In other situations the program inserts the letter sign automatically where necessary (where one has not already been inserted in the input).

Number Sign (#)

The number sign should be punched instead of a period where there may be confusion with a decimal point and the number sign is required in braille. This applies to currency and times of the day, e.g. L6#99, 10#30 A.M.

Grade Switch ($G)

An occurrence of the grade switch, $G, changes the mode of translation from grade 2 to grade 1 or vice versa. It must precede and follow any sequence of characters in which no contractions are to be used, such as a foreign word.

Division Symbol ($/)

The division symbol is the most useful and versatile tool of the braille editor, although its operation is simple: it merely prevents contraction of a letter group which crosses it. For example, for the lisped word "thentury", the "the" contraction is avoided by inserting the division symbol after the "th", thus: TH$/ENTURY. It may be placed between the parts of compound words, such as NUT$/HATCH. It may be placed between prefixes and stems, as in BI$/NOMIAL, or indicate syllable division, as in PERITO$/NEUM and SKI$/DADDLE.

It must be used in a time interval after the hyphen separating minutes from hours, as in 9#30 - $/10#30, in order to produce a number sign correctly.

The division symbol may also be used in cases where the symbols to be translated are coincidentally DOTSYS III control symbols. For example, $$/TAB22 will be translated as "$TAB22" literally, avoiding the control function "tab to column 22." Even "$/" may be generated for output by supplying "$$//" as input.

Forced-Contraction Symbols (/_, _/)

These symbols are used to force a contractible letter group to be contracted. See "Text Input" for a complete description.

OUTPUT

Proof Output

Proof output is optional (see "Run Control Cards"), and is not normally called for in production runs. It is essentially a printout for the use of a sighted braille editor; the braille signs are represented as printed dots. This printout occurs on the same logical device as the error messages (q.v.); error messages may be interspersed with the normal proof output.

If table-display is called for, the first pages of proof output display the contents of the right context table, the decision table, and the transition table in an easily readable form.

The remainder of the output is primarily a page-by-page, line-by-line representation of the braille output, using periods for braille dots. Below the braille sign are up to three characters intended to help identify the sign. If the sign represents a letter, for example, the letter itself is printed below the sign. The identification characters depend only on the sign (as determined by the sign table, q.v.) and not its context. For example, the sign for "K" has the identification character "K" even when it represents the word "knowledge". Below the identification characters is printed the braille sign code in the range 0 to 63. Figure 2 contains a sample of proof output.

A literal listing of the text input cards, as read, is also produced between the lines of braille output (where a "braille output line" is actually a group of five printed lines). These card images will generally occur somewhat sporadically, with two or more occurring together at times and none between pairs of braille lines in other cases. (For separation, a blank line is printed in the latter case.) Typically, a given input word and the corresponding output are separated by about two braille lines.

```
 . .  .. .. .  .     .  .  .     .     .
.        .  .   .    .  .. .     .   .  .
.     .  .  .           .  .     .  .
S  A  M  P  L  E     B  R  L     L  IN E
14 01 13 15 07 17 00 03 23 07 00 07 20 17 00 00 00 00 00
```

Figure 2. Sample of Proof Output

Error Messages

Error messages are unconditionally printed on the proof-output device, thereby appearing intermixed with proof output, if any. The following is a list of the possible messages. For each entry, "A" is an explanation, "B" is the program's automatic action, and "C" is the suggested user response.

(1) NEW CHAR X

A. The character printed (X) has appeared in the input text but does not have an alphabet table entry.

B. The character is replaced by an asterisk and processing proceeds.

C. Correct the text input or provide an alphabet table entry.

(2) ** TABLE SEQUENCE ERROR

A. A contraction table section is out-of-sequence with respect to the alphabet table order.

B. Reading of the tables is continued but processing of the text is suppressed.

C. Refer to "Input" for a discussion of the relationship between the alphabet table arrangement and that of the contraction table, and rearrange accordingly.

SECTION IV

CONTRACTION TABLE SEARCH ALGORITHM

General Rationale

A form of tree search has been implemented for comparison of a string against the contraction table. This algorithm is inherently much more efficient than a linear search for any size table. Moreover, the proportionate cost (in time) for new entries is reduced: the time increases only logarithmically, rather than directly.

The method takes advantage of the fact that often a comparison failure on, say, the third character implies that quite a few additional entries (with the identical first three characters) can be skipped around. The method also conserves space in that only one numeric field must be added to each table entry to support the algorithm.

The basic idea is quite simple. In a given initial-letter section of the table, all the entries which start a new subsection (wherein the first two letters are identical) are chained together, using the added "branch" field as a forward link. This forms the "level 1" chain; a level 2 chain may be formed and so forth. The implicit structure is that of a binary tree. During search, one either follows the branch (continued on the current chain) if comparison failure occurs at the character whose index equals the current level, or else goes to the next entry and one level higher.

The main complication with this process is that, having reached the highest point in the tree and still not having found a match, it still may be necessary to return to the lower level. For example, "AB" may be at level 2 and would probably follow "ABCDE"--at, say, level 4--in the table. The key "BCDF" would thus reach at least level 4 and yet may not actually match until the "AB" is encountered. The handling of this situation in the algorithm is facilitated by indicating the level of the next entry whenever the current level chain ends. This level is entered in the branch field, in negative form so as not to be confused with a forward pointer and also to indicate that the forward pointer is null.

It should be noted that the table must be properly ordered on input in order for the algorithms to work. The proper ordering condition is that entries whose first p letters correspond must be together in the table; formally, using the notation defined below: if there exist two entries, with indices q and r $(q < r)$ and an integer $p > 0$ such that $C_{qj} = C_{rj}$ for all j in the range $1 \leq j \leq p$, then for all t in the range $q \leq t \leq r$ it is also true that $C_{qj} = C_{tj}$ for all j in the range $1 \leq j \leq p$.

Notation

The notation used in the flow charts (Figures 3 and 4) is as follows:

LET

C_{ij}	be the j^{th} character in the i^{th} entry of the contraction table.
b_i	be the value of the branch field for the i^{th} entry.
s_k	be the index of the first entry in the contraction table for the k^{th} initial letter.
e_k	be the index of the last entry for the k^{th} initial letter.
L	be the index for the lowest entry to be considered at the current level.
H	be the index for the highest entry to be considered at the current level.
V	be the highest index for the current initial letter.
n	be the current level.
A_n	be the upper index bound for the n^{th} level.
m	be the number of entries in the contraction table, not counting the extra "end of table" entry.
M	be the number of initial letters.
w_j	be the j^{th} character in the current input string (string being looked up).

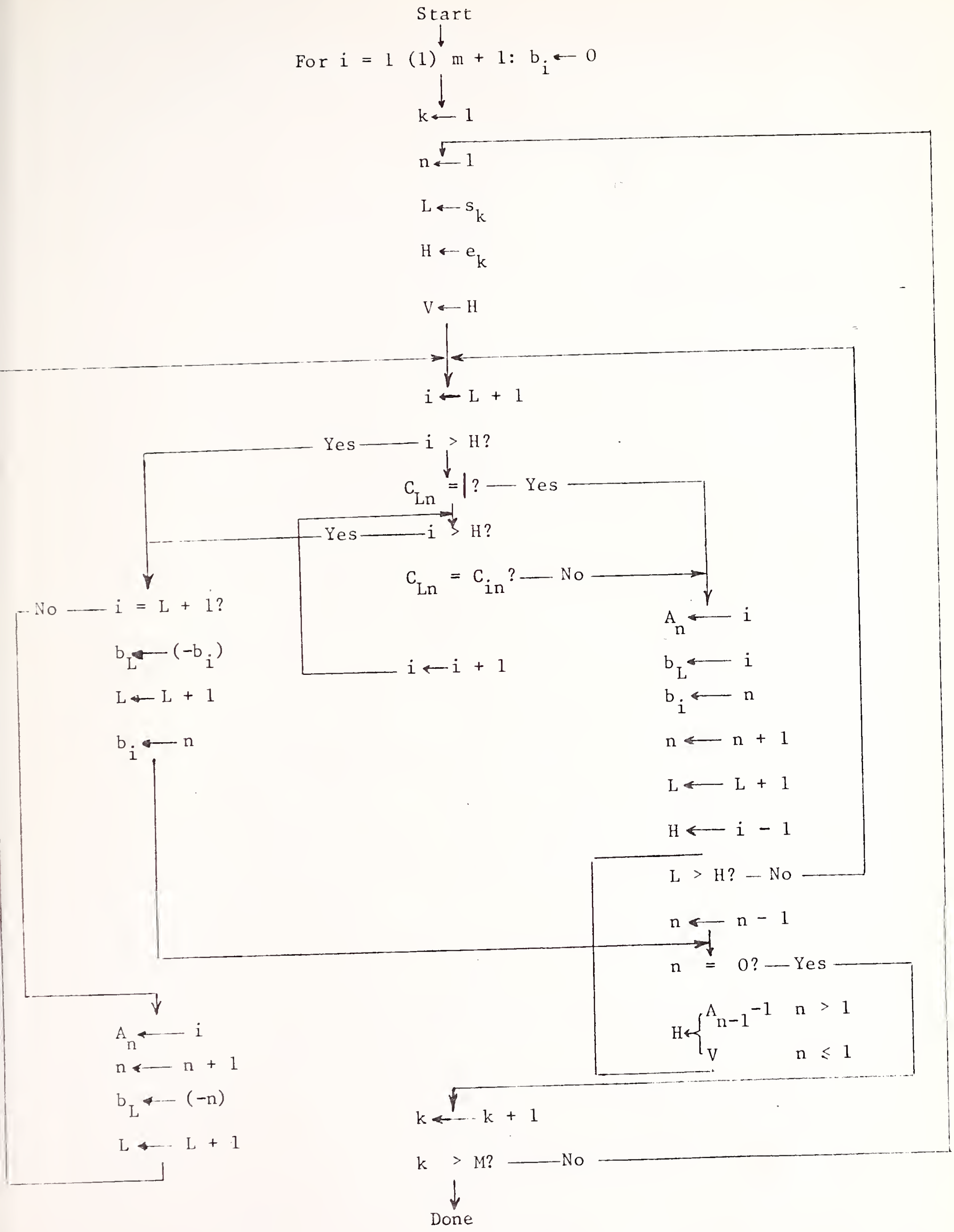


Figure 3. Set-Up Algorithm

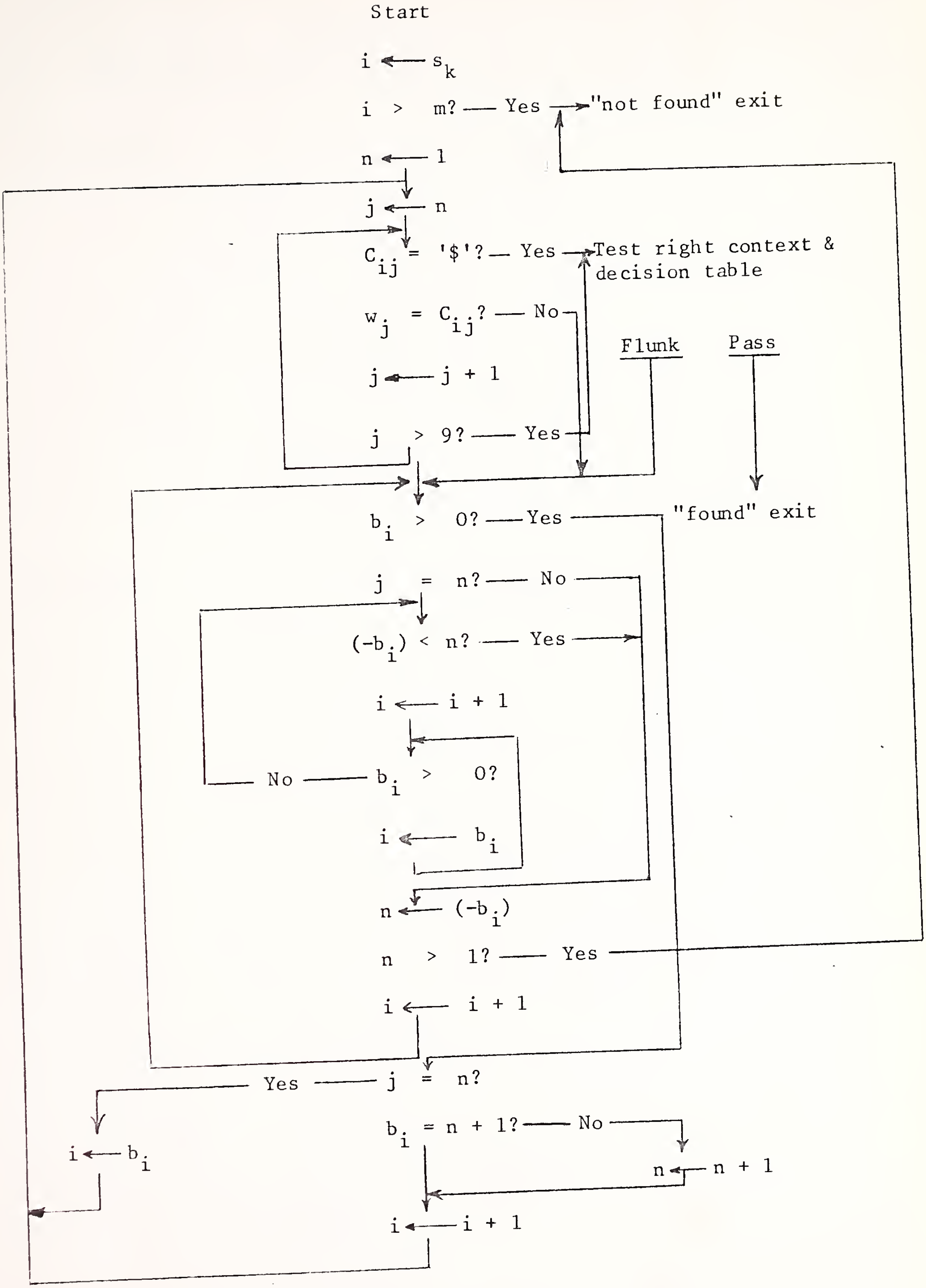


Figure 4. Look-Up Algorithm

APPENDIX 1.1

DEFINITION OF STATE VARIABLES AND INPUT CLASSES

State Variable	1	after the start of a number
	2	after the start of a word
	3	grade 1 translation
	4	in a quotation
	5	in italicized text
	6	not at the start of a prefix or stem
	7	part way through a word or phrase too long for one entry
	8	just after a space (or A-J), following a number
Input Class	1	contractions always used in grade 2
	2	digits
	3	most punctuation
	4	contractions used after the start of a word
	5	$G (grade switch)
	6	contractions used at the start of a word
	7	isolated full-word contractions
	8	$P" (start paragraph in quotation)
	9	$P (start paragraph in italics)
	10	" (left quote)
	11	" (right quote)
	12	_ _ (begin italics)
	13	_ (last word of italics)
	14	(space)

APPENDIX 1.2

SUMMARY OF SPECIAL SYMBOLS

Symbol	Input Representation
capitalize letter	¬
capitalize word	¬¬
italicize word	_
italics start	_ _
left single quote	$'
right single quote	$'R
left double quote	$"
right double quote	$"R
accent marks	¢
left bracket ([)	<
right bracket (])	>
start paragraph	$P
begin new line	$L
begin new page	$PG
skip to column nn	$TABnn
letter sign	=
number sign	#
change grade	$G
divide word	$/
long vowel sign	$LV
start paragraph within quotation	$P"
start title	$TLS

Symbol	Input Representation
end title	$TLE
forced blank	$B
start heading	$HDS
end heading	$HDE
skip lines	$$Lnn
short vowel sign	$SV
set tab t to col. nn	$STBt[L,R or D]nn
end of poetry foot sign	$FT
computer braille switch	$C
caesura sign	$CS
call (permanent) tab t	$#t
start forced contraction	/_
end forced contraction	_/
introductory poetry sign	$PY
poetry sign	$PS

APPENDIX 1.3

EQUIVALENT SIGNS, SYMBOLS AND CODES

	0	1	2	3	4	5	6	7
0	⠀ 00	⠈ c (@) 08	⠐ 5 (") 16	⠘ 45 24	⠠ ¬ 32	⠨ _ . 40	⠰ = ; 48	⠸ 456 _ 56
1	⠁ A 01	⠉ C 09	⠑ E 17	⠙ D 25	⠡ CH * 33	⠩ SH % 41	⠱ WH : 49	⠹ TH ? 57
2	⠂ , 1 02	⠊ I 10	⠒ : 3 18	⠚ J 26	⠢ EN 5 34	⠪ OW , 42	⠲ . 4 50	⠺ W 58
3	⠃ B 03	⠋ F 11	⠓ H 19	⠛ G 27	⠣ GH < 35	⠫ ED $ 43	⠳ OU (\) 51	⠻ ER (]) 59
4	⠄ ’ 04	⠌ ST / 12	⠔ IN 9 20	⠜ AR > 28	⠤ - 36	⠬ ING (+) 44	⠴ " 0 (zero) 52	⠼ # 60
5	⠅ K 05	⠍ M 13	⠕ O 21	⠝ N 29	⠥ U 37	⠭ X 5	⠵ Z 53	⠽ Y 61
6	⠆ ; 2 06	⠎ S 14	⠖ TO 6 22	⠞ T 30	⠦ ? 8 38	⠮ THE ! 46	⠶)(7 54	⠾ WTH (62
7	⠇ L 07	⠏ P 15	⠗ R 23	⠟ Q 31	⠧ V 39	⠯ AND & 47	⠷ OF) 55	⠿ FOR = 63

second octal digit

first octal digit		
	Braille sign	
	Proof character(s)	Computer braille graphic
	Sign code	

FORMAT

Included only where different from proof character. Parentheses around the computer braille graphic indicate that it is not implemented in the tables listed in Appendix I (and will be translated as a blank)

APPENDIX 1.4

TRANSLATOR - STACKER SIGN CODES

Code	Meaning
00	required blank
01-63	braille sign codes
64	end of braille word (blank or end of line)
65	new line
66	unused
67	paragraph start
68	one-time tab (skip to col. nn)
69	new page
70	unused
71	skip (multiple) lines
72	tab (skip according to permanent tab)
73-80	unused
81	start heading input
82	end heading input
83	initialise state-variables and page-numbering
84	set continuous text stacking mode
85	idle (reserved for: reset continuous text stacking mode)
86	unit of measure
87	unused
88	start running title input
89	end of running title input
90	switch double-spacing of lines on or off
93	set tab
95	set margin

Code	Meaning
97	start computer-braille
98	end of run
99	(filler in contraction table)

APPENDIX 1.5

MISCELLANEOUS CODED VARIABLES

Variable	Code	Meaning
STACK-INDICATOR	2	Normal text; clear stack after each entry
	4	Continuous stacking of title or heading
	5	Continuous stacking of normal text
CURRENT-TYPE	1	Normal text or heading stack
	2	Title stack

Note: logical indicator variables are generally coded according to the convention "off" = "no" = "false", "on" = "yes" = "true".

APPENDIX 1.6

RJE TAPE FILE HANDLING PROGRAM

This program incorporates a modified version of DOTSYS III-F which by the inclusion of assembly code achieves a considerably faster translation speed and uses less core storage than the pure Fortran version, together with several subprograms concerned with preparation of data for DOTSYS and embossing of the braille output from DOTSYS.

The program operates on a magnetic tape containing files in Remote Job Entry (RJE) format. The files on this tape may be edited on a visual display unit using the interactive utility text-editing program TED. (See "Summary of Text Editor commands".)

The specific functions of the RJE tape file handling program are as follows:

1. to create a tape file from punched cards
2. to list one or more tape files on the lineprinter
3. to convert the DOTSYS tables (which are stored as a file on the RJE tape) and one or more files of text to card-image format for processing by DOTSYS
4. to store files on a separate archive tape for future reference so that they may be deleted from the RJE tape once they have been translated
5. (DOTSYS) to produce a file containing the braille translation of one or more files of text
6. to emboss a file containing braille on the MIT Braillemboss.

Summary of Text Editor (TED) Commands

The symbols < and > are used below for the purpose of description only and are not actually typed. Underlinings show which character sequences are typed by the user. <CR> refers to the carriage-return key.

File handling commands

```
#BUILD<file name><CR>
  <line 1><CR>
  <line 2><CR>
        .
        .
        .
        .
  <line m><CR>
  <CR>
```

creates a new file with a name given by the string <file name>; the file will contain the lines following the command. Termination of the file content and the BUILD command occurs if two consecutive new-lines are requested.

```
#LIST <file name><CR>
```

will be followed by the listing of the entire named file. This command may be used without an argument, then the names of all files on the loaded tape will be listed.

```
#DELETE <file name><CR>
```

Deletes the named file.

```
#EDIT <file name><CR>
```

Calls in the edit overlay and prepares the named file for editing.

```
# BYE
```

causes the computer to execute some housekeeping on the user files then releases the program. The terminal must not be used again until the program is released, as indicated by a message on the operator's console.

Edit Subcommands

Find commands

```
*FI<integer><CR>
```

Find the line <integer> further down the text than the current line of interest and modify the pointer to create this new line of interest.

```
*FI,<text string><CR>
```

Find the next line that begins with text string and set the pointer to indicate this new line. If the end of the file is reached before the line is found, the search continues from the beginning of the file.

```
*FI<CR>
```

Sets the pointer to the end of the file.

Delete commands

```
*DE<integer><CR>
```

Delete <integer> lines - the current line and <integer>-1 subsequent lines - the pointer is set to the line immediately following the deletion.

```
*DE,<text string><CR>
```

Deletes all lines up to and including the line beginning with <text string>, deletes the current line and moves the pointer to the line following the delection.

```
*DE<CR>
```

Deletes the current line and increments the pointer by one.

Insertion commands

```
*IN CR
<line 1><CR>
<line 2><CR>
     :
```

```
    .
    .
    .
<line m><CR>
<CR>
```

Inserts the lines following the command before the line of current interest; the pointer is not altered. This command is terminated by two consecutive carriage return characters.

Typing commands

```
*TY<integer><CR>
```

Types the current line and the following <integer>-1 lines; the pointer is not changed.

```
*TY,<text string><CR>
```

Types all lines between and including the current line and the line beginning with <text string>; the pointer is unaltered.

```
*TY<CR>
```

Types the current line.

Search limiting commands

```
*LT<integer><CR>
```

Set the limits of searching to <integer> lines. From the current line to the (<integer>-1)th following line.

```
*LT,<text string><CR>
```

Set the limits of searching from the current line to the line beginning with <text string> inclusive.

```
*LT<CR>
```

Reset the limits to the entire file and sets the pointer to the first line in the file.

Substitute command

```
*SU,<text string 1>\<text string 2><CR>
```

Replace the first occurrence of <text string 1> on the current line by <text string 2>. If <text string 1> is omitted, <text string 2> will be inserted at the beginning of the line. If <text string 2> is omitted, <text string 1> will be deleted.

Repeat command

```
*RE<CR>
```

Repeat the last typed editing subcommand.

End of Edit Commands

```
*END<CR>
```

Terminate the edit, updates the file according to the commands given and returns control to the TED command decoder.

```
*<INTERRUPT>
```

terminates the edit and leaves the user file unaltered - as though the most recent sequence of edit commands had not been sent.

Compound commands

Insert and delete

```
*ID<integer><CR>
<line 1><CR>
<line 2><CR>
<line 3><CR>
      .
      .
      .
<line m><CR>
<CR>
```

Deletes <integer> lines including the current line and inserts the lines following the text in that place; the pointer indicates the

line after the insertion on completion of the command. The command is terminated by two consecutive carriage return characters.

```
*ID,<text string><CR>
<line 1><CR>
<CR>
```

Similar to the previously described command but deletion is up to and including the line beginning with <text string>.

```
*ID<CR>
<line 1><CR>
<line 2><CR>
      .
      .
      .
<line m><CR>
<CR>
```

Deletes the current line and inserts the text following the command.

Find and type commands

```
*FT<integer><CR>
```

Sets the pointer <integer> lines further down the text and types the line when it is found.

```
*FT,<text string><CR>
```

Sets the pointer to the next line beginning with <text string> and types the entire line.

NOTES

1. The 'BREAK' or 'INTERRUPT' character halts any listing after a further line. The listing may be terminated by pressing 'INTERRUPT' again, or continued by typing <CR>.
2. In an EDIT command a '?' is 'wild' and can be substituted for any character.
3. 'DEL' deletes the last letter typed in; 'CAN' deletes the current line.

APPENDIX 1.7

USE OF MACROS IN CARD INPUT TO RJE TAPE

(these can save time in typing documents in which a phrase or long word occurs many times.)

A MACRO DEFINITION has the following form:

```
@@<character-pair>=<text>@
```

Where	<character-pair>	is any pair of characters not including an @ sign
and	<text>	is any sequence of characters (of any length) not including an @ sign.

e.g.

```
@@CV=$P COVENTRY (POOL MEADOW)@
@@LL=$P LONG LAWFORD (GREEN)@ @@BT=BRETFORD TURN@
@@XX=$P BRETFORD TURN 05 $P CHURCH LAWFORD (CROSSROADS) 09 $P
LONG LAWFORD (GREEN) 13 $P RUGBY (SHEEP STREET) 20@
@@RC=$P RUGBY CENTRAL@
```

Any number of macro definitions may occur in a document and they may occur anywhere in the input without affecting adjacent text.

A MACRO has the following form:

```
@<character-pair>
```

Where	<character-pair>	is the same as in one of the macro-definitions in the same document.

A macro may occur any number of times anywhere in the input (except in a macro definition) after it has been defined. Each time a macro occurs, the program which copies the cards to tape will replace it with the corresponding text.

e.g.

```
@CV 0805 @LL 15 $P @BT 20 ...
```

or

```
@CV 1300 @XX @RC 25 ....
```

```
C     PRØGRAM-TITLE:  DØTSYS III-F
C     ---------------------------
C
C     THIS PRØGRAM IS A TRANSLATIØN, WITH MØDIFICATIØNS, INTØ RXDS EXTENDED
C     FØRTRAN IV-H ØF THE CØBØL PRØGRAM DØTSYS III.
C     REFERENCE:  DØTSYS III, A PØRTABLE PRØGRAM FØR GRADE II BRAILLE
C     TRANSLATIØN, BY W.R. GERHART, J.K. MILLEN, AND J.E. SULLIVAN,
C     MITRE TECHNICAL REPØRT MTR-2119, THE MITRE CØRPØRATIØN, BEDFØRD,
C     MASSACHUSETTS.
C
C
C     MØDIFICATIØNS MADE TØ ØRIGINAL PRØGRAM
C     --------------------------------------
C
C     THE FØLLØWING HAVE BEEN ØMITTED:
C     SELF-CHECKING AND ALPHABETICAL CØNTRACTIØN LIST; ECHØ ØF TABLES; DØUBLE
C     SPACE PERMITTED AFTER FULL STØP; ØCTAL BRAILLE; ELASTØMER BRAILLE
C     ØUTPUT; PPQ BRAILLE ØUTPUT; PUNCHED CARD ØUTPUT; LINE BY LINE PØETRY.
C
C     THE FØLLØWING HAVE BEEN MØDIFIED:
C     CØMPUTER BRAILLE NØT TERMINATED BY A SPACE; TABLE FØRMAT: RIGHT CØNTEXT
C     CLASS DESIGNATØR IS IN CØLUMN 13 INSTEAD ØF 16 IN CØNTRACTIØN TABLE AND
C     RIGHT CØNTEXT CLASS INPUT.
C
C     THE FØLLØWING HAVE BEEN ADDED:
C     DØUBLE-SPACING ØF ØUTPUT; MARGIN INDENTATIØN; RESET FACILITY FØR
C     PRØCESSING SEPARATE DØCUMENTS CØNSECUTIVELY.
C
C
C     USE ØF VARIABLES
C     ----------------
C
C     ALPHABET TABLE
C     SYMBØL   ACCEPTABLE INPUT CHARACTERS
C     CCLASS   CHARACTER CLASS FØR SYMBØL
C     SINGSG   SINGLE-SIGN TRANSLATIØN FØR SYMBØL
C     EXTENT   INDEX ØF LAST CØNTRACTIØN TABLE ENTRY STARTING WITH GIVEN SYMBØL
C     CØMPB    CØMPUTER BRAILLE TRANSLATIØN FØR SYMBØL
C     ISØLET   "IS-RØUND-LETTER"
C     NALPH    NUMBER ØF ALPHABET TABLE ENTRIES
```

```
C
C     CØNTRACTIØN TABLE
C     CTABLE   CØNTRACTIØN TABLE.  EACH CØLUMN CØNTAINS THE CHARACTER-STRING IN
C              BYTES 1-9 AND THE RIGHT CØNTEXT CLASS IN BYTE 12
C     ICLASS   THE INPUT CLASSES FØR THE CØNTRACTIØN TABLE ENTRIES
C     SIGNS    SHIFT CØUNT AND FØUR BRAILLE SIGN CØDES FØR EACH CØNTRACTIØN
C                 TABLE ENTRY, PACKED ARITHMETICALLY INTØ A SINGLE INTEGER
C     BRANCH   INDEX ØF NEXT ENTRY TØ BE EXAMINED IN CØNTRACTIØN TABLE SEARCH
C     NCT      NUMBER ØF CØNTRACTIØN TABLE ENTRIES
C     CTMAX    ANY NUMBER GREATER THAN THE DIMENSIØN ØF CTABLE
C
C     LØGIC TABLES
C     DECIS    DECISIØN TABLE
C     CØND     CØNDITIØN TABLE
C     TRANS    TRANSITIØN TABLE
C     NØNTER   RIGHT-CØNTEXT DESIGNATØRS
C     RCCLAS   INPUT CLASS CØDES CØRRESPØNDING TØ NØNTER
C     NDTC     NUMBER ØF DECISIØN TABLE ENTRIES
C     NIC      NUMBER ØF INPUT CLASSES
C     NNT      NUMBER ØF NØN-TERMINAL ENTRIES (RIGHT CØNTEXT CLASSES)
C     STVAR    STATE VARIABLES
C     NSV      NUMBER ØF STATE VARIABLES
C
C     STACKS
C     CURTYP   TYPE ØF STACK IN CURRENT USE:
C              1 = NØRMAL TEXT ØR HEADING,  2 = RUNNING TITLE
C     CS       PØINTER TØ RØW ØF CURRENT STACK CURRENTLY IN USE
C     HS       PØINTER TØ HEAD ØF CURRENT STACK
C     TS       PØINTER TØ TAIL ØF CURRENT STACK
C     CURRS    PØINTERS TØ CURRENT RØWS ØF STACKS
C     HEADS    PØINTERS TØ HEADS ØF STACKS
C     TAILS    PØINTERS TØ TAILS ØF STACKS
C     STKIND   STACK INDICATØR, TAKES ØNE ØF THE FØLLØWING VALUES:
C              2   NØRMAL TEXT; CLEAR STACK AFTER EACH ENTRY
C              4   CØNTINUØUS STACKING ØF TITLE ØR HEADING
C              5   CØNTINUØUS STACKING ØF NØRMAL TEXT
C     HFSTAK   PØINTER TØ HEAD ØF FREE STACK
C     ELT      19 RØWS ØF STØRAGE FØR STACKS
C     SPCØNT   SPECIAL CØNTRØL SIGN IF ANY FØR GIVEN RØW ØF STACK
C     NØELTS   NUMBER ØF SIGNS HELD IN GIVEN RØW ØF STACK
```

```
C     PVSTAK   BACKWARD POINTER TO PREVIOUS ROW IN STACK (O FOR FIRST ROW)
C     NXSTAK   FORWARD POINTER TO NEXT ROW IN STACK (O FOR LAST ROW)
C
C     SIGN TABLE
C     DOTS14, DOTS25, DOTS36   BRAILLE SIGN FOR LINE-PRINTER PROOF OUTPUT
C              (SPACE OR DOT)
C     PCHAR    UP TO THREE CHARACTERS ASSOCIATED WITH SIGN FOR PROOF OUTPUT
C     MCODE    SINGLE CHARACTER FOR OUTPUT TO EMBOSSER
C
C     TABULATION
C     TABTYP   TYPES OF TABS (LEFT, RIGHT OR DECIMAL JUSTIFIED)
C     TABCLM   COLUMNS ON WHICH TABS ARE BASED
C     TABULA   COLUMN AT WHICH TABULATED TEXT IS TO BEGIN IN GIVEN CASE
C
C     EMBOSSER OUTPUT
C     EMBOUT   OUTPUT BUFFER FOR EMBOSSER OUTPUT
C     MEMBCR   CARRIAGE-RETURN CHARACTER FOR EMBOSSER
C     MEMBPG   PAGE-FEED CHARACTER FOR EMBOSSER
C     MIDLE    IDLE CHARACTER FOR EMBOSSER
C
C     PROOF OUTPUT
C     BRAILL   OUTPUT BUFFER FOR PROOF BRAILLE SIGNS
C     PROOF    OUTPUT BUFFER FOR PROOF CHARACTERS
C     CODE     OUTPUT BUFFER FOR CODE NUMBER ASSOCIATED WITH BRAILLE SIGN
C     BTEMP, CTEMP, PTEMP   TEMPORARY STORAGE FOR PROOF OUTPUT BUFFER DURING
C              OUTPUT OF PAGE HEADING
C     CPRINT   TRUE IF INPUT CARD IMAGE HAS BEEN PRINTED SINCE LAST PROOF-LINE
C
C     INPUT
C     TEXT     INPUT BUFFER
C     INPTR    POINTER TO CURRENT POSITION IN INPUT BUFFER
C     PRIORS, SCOUNT   USED FOR CONDENSING CONSECUTIVE SPACES IN INPUT
C
C     OUTPUT
C     OUTPTR   POINTER TO CURRENT POSITION IN OUTPUT LINE
C     OUTL     NUMBER OF CHARACTERS PER LINE
C     LPG      NUMBER OF LINES PER PAGE
C     LCOUNT   CURRENT LINE
C
C     LOGICAL VARIABLES
```

```
C      CØMPUT   CØMPUTER BRAILLE TRANSLATIØN
C      EMBØPT   EMBØSSER ØUTPUT REQUIRED
C      PFØUT    PRØØF ØUTPUT REQUIRED
C      PAGINA   PAGINATIØN REQUIRED
C      DØUBLE   DØUBLE SPACING REQUIRED (BLANK LINE BETWEEN EACH LINE ØF ØUTPUT)
C      INHEAD   IN HEADING
C
C      GENERAL
C      CDIGIT   FØUR DIGITS ØF CURRENT PAGE
C      DIGIT    BRAILLE SIGN CØDES FØR DIGITS
C      CCHIGH   HIGHEST VALUE ØF CARRIAGE CØNTRØL CØDES
C      SCHIGH   HIGHEST VALUE ØF STACK CØNTRØL CØDES
C
C
       IMPLICIT INTEGER(A-Z)
       CØMMØN /BUFF/ BUFFER(10)
       EQUIVALENCE (BUFFER(1),BUFF1),(BUFFER(2),BUFF2,RBUFF(1))
       DIMENSIØN RBUFF(9)
       CØMMØN /CTAB/ CTABLE(3,1000),ICLASS(1000),SIGNS(1000),
      *   BRANCH(1000),NCT,TSIGN(4)
       CØMMØN /ALPH/ SYMBØL(64),CCLASS(64),SINGSG(64),EXTENT(64),
      *   CØMPB(64),ISØLET(64),NALPH
       CØMMØN /LØGIC/ DECIS(15,25),CØND(15,10),TRANS(10,25),NØNTER(2),
      *   RCCLAS(4,2),NDTC,NIC,NNT
       CØMMØN /STV/ STVAR(10),NSV
       CØMMØN /SPAC/ SPACE
       CØMMØN /S/ LETS
       CØMMØN /N/ LETN
       CØMMØN /CØMP/ CØMPUT
       LØGICAL CØMPUT
       CØMMØN /RCC/ RCC
       CØMMØN /FILES/ TBDEV,INDEV,PRDEV,BRDEV
C      FØLLØWING CØMMØNS NØT USED IN MAIN PRØG BUT NEEDED FØR ØVERLAY
       CØMMØN /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
      *,HFSTAK,ELT(19,30),SPCØNT(19),NØELTS(19),PVSTAK(19),NXSTAK(19)
       CØMMØN /IN/ INPTR,PRIØRS,SCØUNT
       CØMMØN /ØUT/ ØUTPTR,ØUTL,LINES,LPG
       CØMMØN /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
       CØMMØN /ØPTS/ EMBØPT,PFØUT
       CØMMØN /SIGNS/DØTS14(64),DØTS25(64),DØTS36(64),PCHAR(64),MCØDE(64)
```

```
      COMMON /MEMR/ EMBOUT(11),MEMBCR,MEMBPG,MIDLE,MEMBON,MEMBOF
      COMMON /CPP/ CPRINT
      COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
      COMMON /LC/ LCOUNT
      LOGICAL EMBOPT,PFOUT,CPRINT
C
      DATA LETA,LETF,LETG,LETI,LETN,LETO,LETR,LETS,LETT,LETY,STAR,
     *   HYPHEN,ITALIC,SPACE,LETSIG,CAPSIG,ACCENT,OPBRAK,CLBRAK/
     *   'A','F','G','I','N','O','R','S','T','Y','*','-','_',' ','=',
     *   '¬','¢','(',')'/
      DATA DIGIT /26,1,3,9,25,17,11,27,19,10/
      DATA TBDEV,INDEV,BRDEV,PRDEV /131,131,132,108/
      DATA RCC / '|' /
    1 FORMAT(' NEW CHAR ',A1)
C
C     SEGLOD LOADS SPECIFIED SEGMENT OF OVERLAY
      CALL SEGLOD(3)
      CALL INIT
      CALL SEGLOD(4)
      CALL SHIFT(10)
      CALL PCLEAR
      CALL RESET
C
C     IDENTIFY LEFTMOST CHARACTER OF BUFFER
  200 DO 400 LETTER=1,NALPH
  400 IF (SYMBOL(LETTER) .EQ. BUFF1) GO TO 600
      IF (BUFF1 .NE. RCC .AND. BUFF1 .NE. STAR) GO TO 9800
      CALL OPSIGN(98)
C     RUN WILL STOP WITH ABOVE CALL
C
  600 IF (ISOLET(LETTER) .NE. 1) GO TO 3200
C
C     INSERT LETTER SIGN BEFORE SINGLE LETTER IN BRACKETS, QUOTES, ETC.
      IF (BUFF1 .EQ. ITALIC .AND. (BUFF2 .EQ. LETA .OR. BUFF2 .EQ. LETI
     *   .OR. BUFF2 .EQ. LETO .OR. BUFF2 .EQ. ITALIC)) GO TO 3200
      N = 1
      K = 1
  800 T = RBUFF(N)
      IF (T .EQ. LETSIG) GO TO 3200
      IF (T .EQ. CAPSIG .OR. T .EQ. ACCENT) GO TO 2200
```

```
      IF (T .EQ. ITALIC) GØ TØ 2400
      DØ 1000 X=1,NALPH
 1000 IF (SYMBØL(X) .EQ. T) GØ TØ 1200
      WRITE(PRDEV,1) T
 1200 IF (CCLASS(X) .NE. 1) GØ TØ 3200
      M = N + 1
      IF (BUFF1 .EQ. ITALIC) GØ TØ 2600
      IF (SYMBØL(LETTER) .EQ. ØPBRAK) GØ TØ 1400
      IF (SYMBØL(LETTER) .EQ. RBUFF(M)) GØ TØ 1600
      GØ TØ 3200
C
 1400 IF (RBUFF(M) .NE. CLBRAK) GØ TØ 3200
 1600 DØ 1800 I=1,K
      RBUFF(M) = RBUFF(N)
      N = N - 1
 1800 M = M - 1
      CALL SHIFT(M)
      BUFF1 = LETSIG
      GØ TØ 200
C
 2200 K = K + 1
 2400 N = N + 1
      GØ TØ 800
C
 2600 DØ 2800 X=1,NALPH
 2800 IF (SYMBØL(X) .EQ. RBUFF(M)) GØ TØ 3000
 3000 IF (CCLASS(X) .EQ. 1) GØ TØ 3200
      IF (N .GT. K) CALL SHIFT(1)
      BUFF1 = LETSIG
      GØ TØ 200
C
C     CØNTRACTIØN TABLE SEARCH
C     INDEX PØINTS TØ CØNTRACTIØN TABLE ENTRY UNDER CØNSIDERATIØN
 3200 INDEX = EXTENT(LETTER)
      IF (INDEX .GT. NCT) GØ TØ 9200
      NXTLEV = 1
C
C     TEST FØR MATCH BETWEEN CURRENT TABLE ENTRY AND BUFFER
 3400 DØ 3600 PØINTR=NXTLEV,9
      T = CHAR(CTABLE,INDEX,PØINTR)
```

```
      IF (T .EQ. RCC) GO TO 5000
 3600 IF (RBUFF(POINTR) .NE. T) GO TO 3800
      GO TO 5000
C
C     MISMATCH; OBTAIN INDEX FOR NEXT ENTRY TO BE COMPARED
 3800 BRI = BRANCH(INDEX)
      IF (BRI .GT. 0) GO TO 4600
      IF (POINTR .NE. NXTLEV) GO TO 4400
 4000 IF (-BRI .LT. NXTLEV) GO TO 4400
      INDEX = INDEX + 1
 4200 BRI = BRANCH(INDEX)
      IF (BRI .LE. 0) GO TO 4000
      INDEX = BRI
      GO TO 4200
C
 4400 NXTLEV = -BRI
      IF (NXTLEV .LE. 1) GO TO 9200
      INDEX = INDEX + 1
      GO TO 3400
 4600 IF (POINTR .EQ. NXTLEV) GO TO 4800
      IF (BRI .NE. INDEX+1) NXTLEV = NXTLEV + 1
      INDEX = INDEX + 1
      GO TO 3400
 4800 INDEX = BRI
      GO TO 3400
C
C     CHARACTER STRING MATCHES LEFT SUBSTRING OF BUFFER
C     NOW TEST NEXT CHARACTER IN BUFFER TO CHECK THAT RIGHT CONTEXT IS O.K.
 5000 T = CHAR(CTABLE,INDEX,12)
      IF (T .EQ. SPACE) GO TO 6200
      DO 5200 N=1,NALPH
 5200 IF (SYMBOL(N) .EQ. RBUFF(POINTR)) GO TO 5400
      GO TO 3800
C
 5400 DO 6000 K=1,NNT
      IF (T .NE. NONTER(K)) GO TO 6000
      DO 5800 J=1,4
      IF (CCLASS(N) .EQ. RCCLAS(J,K)) GO TO 6200
 5800 IF (RCCLAS(J,K) .EQ. 99) GO TO 3800
      GO TO 3800
```

```
 6000 CØNTINUE
      GØ TØ 3800
C .
C     RIGHT CØNTEXT IS Ø.K. - NØW CHECK INPUT CLASS FØR CØMPATIBILITY
C     WITH CURRENT VALUE ØF STATE VARIABLES
 6200 TCLASS = ICLASS(INDEX)
      DØ 7000 K=1,NDTC
      DKT = DECIS(K,TCLASS)
      IF (DKT .EQ. HYPHEN) GØ TØ 7000
      IF (DKT .EQ. LETG) GØ TØ 7600
      IF (DKT .EQ. LETF) GØ TØ 3800
      DØ 6800 N=1,NSV
      CKN = CØND(K,N)
 6800 IF (CKN .NE. HYPHEN .AND. CKN .NE. STVAR(N)) GØ TØ 7400
      GØ TØ 7200
 7000 CØNTINUE
C
 7200 IF (DKT .EQ. LETY) GØ TØ 7600
      GØ TØ 3800
 7400 IF (DKT .EQ. LETY) GØ TØ 3800
C
C     CØNTRACTIØN TABLE ENTRY IS ACCEPTED
C     SEND APPRØPRIATE SIGNS TØ STACKER AND SHIFT BUFFER LEFT BY
C     NUMBER ØF CHARACTERS INDICATED IN CØNTRACTIØN TABLE ENTRY
 7600 T1 = SIGNS(INDEX)
      T = T1 / 11
      CALL SHIFT(T1 - 11 * T)
      DØ 8200 J=1,4
      T1 = T / 100
      ØUTSIG = T - 100 * T1
      IF (ØUTSIG .EQ. 99) GØ TØ 8400
      T = T1
 8200 CALL ØPSIGN(ØUTSIG)
C
C     ADJUST STATE VARIABLES ACCØRDING TØ TRANSITIØN TABLES
 8400 I = TCLASS
      DØ 9000 N=1,NSV
      TNI = TRANS(N,I)
      IF (TNI .EQ. HYPHEN) GØ TØ 9000
      IF (TNI .NE. LETS .AND. (TNI .NE. LETT .ØR. STVAR(N) .NE. LETN))
```

```
     *    GØ TØ 8800
      STVAR(N) = LETY
      GØ TØ 9000
 8800 STVAR(N) = LETN
 9000 CØNTINUE
      GØ TØ 200
C
C     NØ CØNTRACTIØN TABLE ENTRY IS ACCEPTABLE - ØUTPUT SINGLE CHARACTER
C     ØBTAINED FRØM ALPHABET TABLE
 9200 IF (.NØT. CØMPUT) GØ TØ 9400
      ØUTSIG = CØMPB(LETTER)
      GØ TØ 9600
 9400 ØUTSIG = SINGSG(LETTER)
 9600 CALL SHIFT(1)
      CALL ØPSIGN(ØUTSIG)
      TCLASS = CCLASS(LETTER)
      GØ TØ 8400
C
C     LEFTMØST CHARACTER ØF BUFFER DØES NØT ØCCUR IN ALPHABET TABLE
C     ØUTPUT ERRØR MESSAGE
 9800 WRITE(PRDEV,1) BUFF1
      BUFF1 = STAR
      GØ TØ 200
      END


      INTEGER FUNCTIØN CHAR(ARRAY,I,J)
C     VALUE ØF CHAR IS CHARACTER IN J TH RØW ØF I TH CØLUMN ØF ARRAY,
C     LEFT JUSTIFIED
      IMPLICIT INTEGER(A-Z)
      DATA ADDEND /6Z404040/
      CHAR = ISL(ICH(ARRAY,12*(I-1)+J),24) + ADDEND
      END


      SUBRØUTINE STCHAR(CHAR,ARRAY,PØSN)
      IMPLICIT INTEGER(A-Z)
C     STØRES LEFT-HAND BYTE ØF CHAR IN PØSN TH BYTE ØF ARRAY
S        LB,9       *CHAR
S        LW,7       *PØSN
```

```
S          AI,7       -1
S          STB,9      *ARRAY,7
         END


         SUBROUTINE INIT
C
C        INIT - INITIALISATION.  READ TABLES AND CONTROL CARDS.  SET UP CONTRACTION
C           TABLE SEARCH POINTERS. SET UP STACK POINTERS, TABS ETC. INITIALISE
C           VARIABLES.
C
         IMPLICIT INTEGER(A-Z)
         COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
        *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
         COMMON /IN/ INPTR,PRIORS,SCOUNT
         COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
         COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
         COMMON /OPTS/ EMBOPT,PFOUT
         COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),PCHAR(64),MCODE(64)
         COMMON /MEMB/ EMBOUT(11),MEMBCR,MEMBPG,MIDLE,MEMBON,MEMBOF
         COMMON /CPR/ CPRINT
         COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
         COMMON /SPAC/ SPACE
         COMMON /CTAB/ CTABLE(3,1000),ICLASS(1000),SIGNS(1000),
        *   BRANCH(1000),NCT,TSIGN(4)
         COMMON /ALPH/ SYMBOL(64),CCLASS(64),SINGSG(64),EXTENT(64),
        *   COMPB(64),ISOLET(64),NALPH
         COMMON /LC/ LCOUNT
         COMMON /LOGIC/ DECIS(15,25),COND(15,10),TRANS(10,25),NONTER(2),
        *   RCCLAS(4,2),NDTC,NIC,NNT
         COMMON /STV/ STVAR(10),NSV
         COMMON /RCC/ RCC
         LOGICAL EMBOPT,PFOUT,SEGERR,PAGINA,CPRINT
         COMMON /FILES/ TBDEV,INDEV,PRDEV,BRDEV
         DIMENSION LARRAY(10)
         DATA LETE,LETL,LETM,LETP,LETT /'E','L','M','P','T'/
         DATA CTMAX /1000/
C
       1 FORMAT(80X)
       2 FORMAT(' ** TABLE SEQUENCE ERROR')
```

```
    3 FØRMAT(A1,71X)
    4 FØRMAT(A1,16X,3(I2,X),3X,I2)
    5 FØRMAT(A1,3A4,4X,2(I2,X),4I2)
    6 FØRMAT(17X,I2)
    7 FØRMAT(12X,A1,10X,4I2)
    8 FØRMAT(25A1)
    9 FØRMAT(NA1,NA1)
   10 FØRMAT(3A2,A3,A1)
   11 FØRMAT(A1,14X,5A1)
   12 FØRMAT(A1,30X,4I1)
   13 FØRMAT(' TØØ MANY CØNTRACTIØN TABLE ENTRIES')
C
      INPTR = 81
      SEQERR = .FALSE.
C
C     SET UP STACKS
      DØ 200 N=1,19
      NØELTS(N) = 0
      SPCØNT(N) = 0
      PVSTAK(N) = N - 1
  200 NXSTAK(N) = N + 1
      NXSTAK(19) = 0
      NXSTAK(1) = 0
      PVSTAK(2) = 0
      HFSTAK = 3
      CURTYP = 1
      HEADS(1) = 1
      TAILS(1) = 1
      CURRS(1) = 1
      HEADS(2) = 2
      TAILS(2) = 2
      CURRS(2) = 2
      HS = 1
      TS = 1
      CS = 1
      PVSTAK(3) = 0
      NXSTAK(2) = 0
C
      DØ 600 N = 1,9
      TABTYP(N) = LETL
```

```
  600 TABCLM(N) = 2 * N + 1
      LCØUNT = 0
      ØUTPTR = 1
      TABLLA = 0
      SCØUNT = 0
      TEMPL = 0
      STKIND = 2
      CPRINT = .FALSE.
      PRIØRS = 0
C
C     READ ALPHABET TABLE
      DØ 800 N=1,65
      EXTENT(N) = 999
      READ(TBDEV,4) SYMBØL(N),CCLASS(N),CØMPB(N),SINGSG(N),ISØLET(N)
  800 IF (CCLASS(N) .EQ. 99) GØ TØ 1000
 1000 NALPH = N
      I = 1
C
C     FILL CØNTRACTIØN TABLE
      DØ 1800 N=1,CTMAX
      READ(TBDEV,5) LET1,(CTABLE(J,N),J=1,3),ICLASS(N),SHIFNØ,
     *   (TSIGN(J),J=1,4)
C     PACK SIGNS AND SHIFT CØUNT INTØ SIGNS(N)
      SIGNS(N) = 0
      DØ 1100 J=4,1,-1
 1100 SIGNS(N) = 100 * SIGNS(N) + TSIGN(J)
      SIGNS(N) = 11 * SIGNS(N) + SHIFNØ
      IF (LET1 .EQ. TEMPL) GØ TØ 1400
      TEMPL = LET1
      EXTENT(I) = N
      IF (SYMBØL(I) .EQ. TEMPL .ØR. ICLASS(N) .EQ. 99) GØ TØ 1200
      SEQERR = .TRUE.
      WRITE(PRDEV,2)
 1200 I = I + 1
 1400 CØNTINUE
 1300 IF (ICLASS(N) .EQ. 99) GØ TØ 2000
C
      WRITE(PRDEV,13)
      STØP 'ERRØR'
C
```

```
 2000 NCT = N - 1
      J = I - 1
      EXTENT(I) = EXTENT(J) + 1
      MAXEXT = J
C
C     CØNTRACTIØN TABLE SEARCH SET-UP ALGØRITHM
      T = NCT + 1
      DØ 2200 I=1,T
 2200 BRANCH(I) = 0
      DØ 4000 I=1,MAXEXT
      N = 1
      LØWNØ = EXTENT(I)
      J = I + 1
      HIGHNØ = EXTENT(J) - 1
      VHIGH = HIGHNØ
 2600 II = LØWNØ + 1
      IF (II .GT. HIGHNØ) GØ TØ 3600
      IF (CHAR(CTABLE,LØWNØ,N) .EQ. RCC) GØ TØ 3000
 2800 IF (II .GT. HIGHNØ) GØ TØ 3600
      IF (CHAR(CTABLE,LØWNØ,N) .NE. CHAR(CTABLE,II,N)) GØ TØ 3000
      II = II + 1
      GØ TØ 2800
 3000 LARRAY(N) = II
      BRANCH(LØWNØ) = II
      BRANCH(II) = N
      N = N + 1
      LØWNØ = LØWNØ + 1
      HIGHNØ = II - 1
 3200 IF (LØWNØ .LE. HIGHNØ) GØ TØ 2600
      N = N - 1
 3400 IF (N .EQ. 0) GØ TØ 4000
      IF (N .LE. 1) GØ TØ 3500
      HIGHNØ = LARRAY(N-1) - 1
      GØ TØ 3200
 3500 HIGHNØ = VHIGH
      GØ TØ 3200
 3600 IF (II .NE. LØWNØ + 1) GØ TØ 3800
      BRANCH(LØWNØ) = - BRANCH(II)
      LØWNØ = LØWNØ + 1
      BRANCH(II) = N
```

```
      GØ TØ 3400
 3800 LARRAY(N) = II
      N = N + 1
      BRANCH(LØWNØ) = -N
      LØWNØ = LØWNØ + 1
      GØ TØ 2600
 4000 CØNTINUE
C
C     READ RIGHT-CØNTEXT TABLE
      READ(TBDEV,6) NNT
      DØ 4200 N=1,NNT
 4200 READ(TBDEV,7) NØNTER(N),(RCCLAS(I,N),I=1,4)
C
C     READ STATE TABLES
      READ(TBDEV,6) NSV
      READ(TBDEV,6) NIC
      DØ 4400 I=1,NSV
 4400 READ(TBDEV,8) (TRANS(I,J),J=1,25)
      READ(TBDEV,6) NDTC
      DØ 4600 N =1,NDTC
 4600 READ(TBDEV,9) NIC,(DECIS(N,I),I=1,NIC),NSV,(CØND(N,I),I=1,NSV)
C
C     READ SIGN TABLE
      DØ 5000 N = 1,64
 5000 READ(TBDEV,10) DØTS14(N),DØTS25(N),DØTS36(N),PCHAR(N),MCØDE(N)
C
C     READ CØNTRØL CARDS
      READ(TBDEV,11) T,T1
      PFØUT = T .EQ. LETP
      IF (PFØUT .AND. T1 .EQ. LETT) CALL TABDIS
      READ(TBDEV,11) T,MEMBØN,MEMBØF,MIDLE,MEMBCR,MEMBPG
      EMBØPT = T .EQ. LETM
      CALL STCHAR(MIDLE,EMBØUT,1)
      CALL STCHAR(MEMBØN,EMBØUT,2)
      CALL STCHAR(MIDLE,EMBØUT,41)
      CALL STCHAR(MEMBØF,EMBØUT,42)
      READ(TBDEV,6) ØUTL
      READ(TBDEV,6) LPG
      READ(TBDEV,12) T,(CDIGIT(I),I=1,4)
      PAGINA = T .EQ. LETP
```

```
      IF (SEQERR) STOP 'SEQUENCE ERROR'
      END

      SUBROUTINE TABDIS
C
C     TABDIS - TABLE DISPLAY FØR LØGIC TABLES
C
      IMPLICIT INTEGER(A-Z)
      CØMMØN /STV/ STVAR(10),NSV
      CØMMØN /LØGIC/ DECIS(15,25),CØND(15,10),TRANS(10,25),NØNTER(2),
     *   RCCLAS(4,2),NDTC,NIC,NNT
      CØMMØN /FILES/ TBDEV,INDEV,PRDEV,BRDEV
   10 FØRMAT(///16X,'DECISIØN TABLE'//'-CØLUMN')
   11 FØRMAT(///16X,'RIGHT CØNTEXT TABLE'//' NØN-TERMINAL    INPUT CLASSE
     *S'//)
   12 FØRMAT(6X,A1,8X,4(I2,2X))
   13 FØRMAT(19X,20(2X,I2))
   14 FØRMAT(/'-INPUT CLASS')
   15 FØRMAT(17X,I2,3X,20(A1,3X))
   16 FØRMAT(/'-STATE VARIABLE')
   17 FØRMAT(///16X,'TRANSITIØN TABLE'//'-STATE VARIABLE')
   18 FØRMAT(//)
      WRITE(PRDEV,11)
      DØ 6200 K=1,NNT
 6200 WRITE(PRDEV,12) NØNTER(K),(RCCLAS(I,K),I=1,4)
      WRITE(PRDEV,10)
      WRITE(PRDEV,13) (K,K=1,NDTC)
      WRITE(PRDEV,14)
      DØ 6400 N=1,NIC
 6400 WRITE(PRDEV,15) N,(DECIS(K,N),K=1,NDTC)
      WRITE(PRDEV,16)
      DØ 6600 N=1,NSV
 6600 WRITE(PRDEV,15) N,(CØND(K,N),K=1,NDTC)
      WRITE(PRDEV,17)
      WRITE(PRDEV,13) (K,K=1,NSV)
      WRITE(PRDEV,14)
      DØ 6800 N=1,NIC
 6800 WRITE(PRDEV,15) N,(TRANS(K,N),K=1,NSV)
      WRITE(PRDEV,18)
```

```
      END

      SUBROUTINE SHIFT(NCHARS)
C
C     SHIFT(NCHARS) - INPUT PROCESSOR.  OBTAINS INPUT FROM INPUT DEVICE, WITH
C        ECHO ON PROOF DEVICE IF PROOF OUTPUT REQUESTED. CONDENSES CONSECUTIVE
C        SPACES. REMOVES ALL OCCURENCES OF RCC CHARACTER (VERTICAL BAR). INSERTS
C        RCC CHARACTER AT END OF INPUT. SHIFTS BUFFER LEFT BY NCHARS AND FILLS
C        RIGHT HAND SIDE WITH RESULT OF ABOVE PROCESS.
C
      IMPLICIT INTEGER(A-Z)
      COMMON /IN/ INPTR,PRIORS,SCOUNT
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /BUFF/ BUFFER(10)
      COMMON /OPTS/ EMBOPT,PFOUT
      LOGICAL EMBOPT,PFOUT
      COMMON /SPAC/ SPACE
      COMMON /CPR/ CPRINT
      LOGICAL CPRINT
      COMMON /RCC/ RCC
      COMMON /FILES/ TBDEV,INDEV,PRDEV,BRDEV
      DIMENSION TEXT(20)
    1 FORMAT(20A4)
    2 FORMAT(X,20A4)
C
      IF (NCHARS .EQ. 0) RETURN
      DO 1400 I=1,NCHARS
      DO 50 N=1,9
   50 BUFFER(N) = BUFFER(N+1)
  100 IF (INPTR .LE. 80) GO TO 200
      IF (SCOUNT .GT. 0) GO TO 800
      READ(INDEV,1,END=800) TEXT
      INPTR = 1
      IF (.NOT. PFOUT) GO TO 200
      WRITE(PRDEV,2) TEXT
      CPRINT = .TRUE.
  200 NXTCHR = CHAR(TEXT,1,INPTR)
      INPTR = INPTR + 1
      IF (NXTCHR .NE. SPACE) GO TO 400
```

```
      IF (PRIØRS .EQ. 0) GØ TØ 100
      PRIØRS = PRIØRS - 1
      GØ TØ 600
  400 PRIØRS = 1
  600 IF (NXTCHR .EQ. RCC) GØ TØ 100
      GØ TØ 1200
  800 SCØUNT = SCØUNT + 1
      IF (SCØUNT .EQ. 1 .AND. PRIØRS .GT. 0) GØ TØ 1000
      NXTCHR = RCC
      GØ TØ 1200
 1000 NXTCHR = SPACE
 1200 BUFFER(10) = NXTCHR
 1400 CØNTINUE
      END


      SUBRØUTINE ØPSIGN(ØUTSIG)
C
C     ØPSIGN(ØUTSIG) - STACKER.  TAKES ØUTSIG AS INPUT. CLEARS CURRENT STACK IF
C        REQUIRED (SPACE, END HEADING, END TITLE), CHANGES CURRENT STACK IF
C        REQUIRED (START ØR END HEADING ØR TITLE), CHANGES LØGICAL VARIABLES IF
C        REQUIRED (DØUBLE-SPACING, CØMPUTER BRAILLE), ØTHERWISE STØRES SIGN ØN
C        CURRENT STACK. INTERCHANGES LAST TWØ RØWS ØF STACK FØR UNITS ØF MEASURE
C
      IMPLICIT INTEGER(A-Z)
      CØMMØN /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCØNT(19),NØELTS(19),PVSTAK(19),NXSTAK(19)
      CØMMØN /ØUT/ ØUTPTR,ØUTL,LINES,LPG
      CØMMØN /PAG/ CDIGIT(4),DIGIT(10),PAGINA
      CØMMØN /BUFF/ BUFFER(10)
      CØMMØN /ØPTS/ EMBØPT,PFØUT
      CØMMØN /S/ LETS
      CØMMØN /TTL/ TTLENG
      CØMMØN /CØMP/ CØMPUT
      CØMMØN /INDENT/ MARGIN,INHEAD
      CØMMØN /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
      CØMMØN /DUB/ DØUBLE
      CØMMØN /FILES/ TBDEV,INDEV,PRDEV,BRDEV
      LØGICAL PAGINA,EMBØPT,PFØUT,INHEAD,CØMPUT,DØUBLE
      DATA CCHIGH,SCHIGH/80,89/
```

```
C
C     CHARACTER TEST
      IF (OUTSIG - 64) 800,400,200
  200 IF (OUTSIG .LE. CCHIGH) GO TO 1400
      IF (OUTSIG .LE. SCHIGH) GO TO 1800
      GO TO 2000
C
C     OUTPUT SPACE
  400 IF (STKIND .NE. 2) GO TO 600
      CALL CLEAR
      RETURN
  600 CALL STAKTB
      RETURN
C
C     OUTPUT ORDINARY CHARACTER
  800 IF (NOELTS(CS) .NE. 30) GO TO 1000
      CALL STAKTB
      NOELTS(CS) = 1
      GO TO 1200
 1000 NOELTS(CS) = NOELTS(CS) + 1
 1200 ELT(CS,NOELTS(CS)) = OUTSIG
      RETURN
C
C     CARRIAGE CONTROL
 1400 IF (NOELTS(CS) .NE. 0) CALL STAKTB
      NOELTS(CS) = 30
      SPCONT(CS) = OUTSIG
      IF (OUTSIG .NE. 72) GO TO 1600
      CALL DECODE(1,ELT(CS,3))
      CALL SHIFT(1)
 1500 IF (STKIND .EQ. 2) STKIND = 5
      RETURN
 1600 IF (OUTSIG .NE. 68 .AND. OUTSIG .NE. 71) RETURN
 1700 CALL DECODE(2,ELT(CS,3))
 1750 CALL SHIFT(2)
      RETURN
C
C     STACK CONTROL
 1800 T = OUTSIG - 80
      GO TO (2400,2600,5000,1500,1900,4600,1900,3600,4000),T
```

```
 1900 RETURN
C
C       MODE CONTROL
 2000 IF (ØUTSIG .EQ. 93) GØ TØ 5800
      IF (ØUTSIG .EQ. 95) GØ TØ 4800
      IF (ØUTSIG .EQ. 98) GØ TØ 5600
      IF (ØUTSIG .EQ. 97) CØMPUT = .NØT. CØMPUT
      IF (ØUTSIG .EQ. 30) DØUBLE = .NØT. DØUBLE
      RETURN
C
C       BEGIN HEADING
 2400 IF (NØELTS(CS) .NE. 0) CALL CLEAR
      CALL CARRIJ(65)
      STKIND = 4
      RETURN
C
C       END HEADING
 2600 N=HEADS(1)
      M = 0
 2800 M = M + NØELTS(N)
      IF (SPCØNT(N) .EQ. 0 .AND. NØELTS(N) .GT. 0) M = M + 1
      IF (N .EQ. TAILS(1)) GØ TØ 3000
      N = NXSTAK(N)
      GØ TØ 2800
C
 3000 ØUTPTR = (ØUTL - M) / 2 + 2
      IF (ØUTPTR .LT. 1) ØUTPTR = 1
      STKIND = 2
      INHEAD = .TRUE.
      CALL CLEAR
      INHEAD = .FALSE.
      CALL CARRIJ(65)
      RETURN
C
C       BEGIN TITLE
 3600 CURTYP = 2
      HS = HEADS(2)
      TS = TAILS(2)
      CS = CURRS(2)
      STKIND = 4
```

```
      M = HS
      IF (NØELTS(M) .EQ. 0) RETURN
C     CLEAR ØUT PREVIOUS TITLE
      NØELTS(M) = 0
 3800 IF (HS .EQ. TS) RETURN
      CALL FRSTAK(PVSTAK,NXSTAK,TS,TAILS)
      CURRS(2) = HEADS(2)
      GØ TØ 3800
C
C     END TITLE
 4000 TTLENG = 0
      N = HEADS(2)
 4200 TTLENG = TTLENG + NØELTS(N)
      IF (SPCØNT(N) .EQ. 0 .AND. NØELTS(N) .GT. 0) TTLENG = TTLENG + 1
      IF (N .EQ. TAILS(2)) GØ TØ 4400
      N = NXSTAK(N)
      GØ TØ 4200
C
 4400 CURTYP = 1
      HS = HEADS(1)
      TS = TAILS(1)
      CS = CURRS(1)
      STKIND = 2
      RETURN
C
C     UNIT ØF MEASURE
 4600 SPCØNT(CS) = 1
C
C     INTERCHANGE LAST TWØ RØWS ØF CURRENT STACK
      N = PVSTAK(TS)
      IF (N .EQ. 0) GØ TØ 4700
      M = PVSTAK(N)
      PVSTAK(TS) = M
      NXSTAK(TS) = N
      PVSTAK(N) = TS
      NXSTAK(N) = 0
      TS = N
      TAILS(CURTYP) = N
      CS = N
      CURRS(CURTYP) = N
```

```
      IF (N .NE. HS) GØ TØ 4650
      HS = PVSTAK(N)
      HEADS(CURTYP) = HS
      GØ TØ 4700
 4650 NXSTAK(M) = PVSTAK(N)
C
 4700 IF (BUFFER(1) .EQ. LETS) CALL SHIFT(1)
      RETURN
C
C     MARGIN RESET
 4800 CALL DECØDE(2,MARGIN)
      CALL SHIFT(2)
      RETURN
C
C     RESTART
 5000 DØ 5200 N=1,3
 5200 CDIGIT(N) = 0
      CDIGIT(4) = 1
      CALL RESET
      RETURN
C
 5600 CALL CARRIJ(65)
      ENDFILE BRDEV
      ENDFILE PRDEV
      STØP 'ØK'
 5800 CALL DECØDE(1,T)
      TABTYP(T) = BUFFER(2)
      CALL SHIFT(2)
      CALL DECØDE(2,TABCLM(T))
      CALL SHIFT(2)
      RETURN
      END


      SUBRØUTINE CARRIJ(ØUTSIG)
C
C     CARRIJ(ØUTSIG) - CARRIAGE CØNTRØL.
C
      IMPLICIT INTEGER(A-Z)
      CØMMØN /ØUT/ ØUTPTR,ØUTL,LINES,LPG
```

```
      COMMON /LC/ LCOUNT
      COMMON /DUB/ DOUBLE
      LOGICAL DOUBLE
C
      IF (LCOUNT .GE. LPG) CALL PAGEHD
      IF (OUTPTR .EQ. 1 .AND. (OUTSIG .EQ. 65 .OR. OUTSIG .EQ. 67))
     *   GO TO 600
      CALL OUTOUT(OUTSIG)
      IF (DOUBLE .AND. LCOUNT .LT. LPG) CALL OUTOUT(OUTSIG)
  600 IF (OUTSIG .NE. 67) GO TO 800
      OUTPTR = 3
      RETURN
  800 OUTPTR = 1
      END


      SUBROUTINE CLEAR
C
C     CLEAR - CLEAR CURRENT STACK: LINE COMPOSER. ROWS OF CURRENT STACK ARE SENT
C        TO OUTPUT BUFFER. PERFORMS TABULATION OR SKIP-LINES WHERE CALLED FOR.
C
      IMPLICIT INTEGER(A-Z)
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
      COMMON /INDENT/ MARGIN,INHEAD
      LOGICAL INHEAD
C
  100 PS = HS
      NPS = NOELTS(PS)
  400 IF (SPCONT(PS) .LT. 64) GO TO 3000
      OUTSIG = SPCONT(PS)
      EP3 = ELT(PS,3)
      IF (OUTSIG .NE. 68) GO TO 600
C     ONE-TIME TABULATION
      TABULA = EP3
      GO TO 4100
  600 IF (OUTSIG .NE. 71) GO TO 1800
C     SKIP LINES
```

```
      CALL CARRIJ(65)
      CALL PCLEAR
      DØ 1600 M=1,EP3
      ØUTPTR = 0
 1600 CALL CARRIJ(71)
      GØ TØ 4200
C
C     PRESET TABULATIØN
 1800 IF (ØUTSIG .NE. 72) GØ TØ 2800
      M = ELT(PS,3)
      XX = NXSTAK(PS)
      IF (TABTYP(M) .EQ. R) GØ TØ 2000
      IF (TABTYP(M) .EQ. D) GØ TØ 2200
      TABULA = TABCLM(M)
      GØ TØ 4100
C
 2000 TABULA = TABCLM(M) - NØELTS(XX) + 1
      GØ TØ 4100
C
 2200 T = NØELTS(XX)
      DØ 2400 XYZ=1,T
 2400 IF (ELT(XX,XYZ) .EQ. 40) GØ TØ 2600
 2600 TABULA = TABCLM(M) - XYZ + 1
      GØ TØ 4100
C
 2800 CALL CARRIJ(ØUTSIG)
      GØ TØ 4200
C
 3000 IF (NPS .EQ. 0) GØ TØ 4200
      IF (ØUTPTR + NPS .LT. ØUTL + 2) GØ TØ 3800
      ØUTPTR = ØUTL + 1
      CALL CARRIJ(0)
      IF (MARGIN .LE. 0) GØ TØ 3400
      DØ 3200 N=1,MARGIN
 3200 CALL MØVEEL(64)
 3400 IF (.NØT. INHEAD) GØ TØ 3800
      DØ 3600 N=1,3
 3600 CALL MØVEEL(64)
 3800 DØ 4000 M=1,NPS
 4000 CALL MØVEEL(ELT(PS,M))
```

```
      IF (SPCONT(PS) .NE. 1) CALL MOVEEL(64)
      GO TO 4200
C
C     TABULATION
 4100 TABULA = TABULA - OUTPTR
      IF (TABULA .LE. 0) GO TO 4120
      DO 4110 N=1,TABULA
 4110 CALL MOVEEL(64)
      GO TO 4140
 4120 TABULA = TABULA + OUTPTR - 1
      CALL CARRIJ(65)
      DO 4130 N=1,TABULA
 4130 CALL MOVEEL(64)
 4140 TABULA = 0
C
 4200 IF (HS .EQ. TS) GO TO 4800
      CALL FRSTAK(NXSTAK,PVSTAK,HS,HEADS)
      GO TO 100
 4800 NOELTS(HS) = 0
      SPCONT(HS) = 0
      CURRS(CURTYP) = HS
      CS = HS
      END


      SUBROUTINE FRSTAK(STAK,STAK1,HT,HTS)
C
C     FRSTAK(STAK,STAK1,HT,HTS) - FREE-STACK. SETS STACK POINTERS TO TRANSFER
C        LAST ROW OF CURRENT STACK TO FREE STACK.
C
      IMPLICIT INTEGER(A-Z)
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      DIMENSION STAK(19),STAK1(19),HTS(2)
C
      NOELTS(HT) = 0
      SPCONT(HT) = 0
      IF (STAK(HT) .EQ. 0) RETURN
      I = HT
      N = HFSTAK
```

```
      HT = STAK(M)
      HTS(CURTYP) = HT
      HFSTAK = M
      STAK1(HT) = 0
      NXSTAK(M) = N
      END


      SUBROUTINE MOVEEL(X)
C
C     MOVEEL(X) - MOVE ELEMENT.  TRANSFERS THE SIGN WHOSE CODE IS X TO THE
C        OUTPUT BUFFERS.
C
      IMPLICIT INTEGER(A-Z)
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),PCHAR(64),MCODE(64)
      COMMON /OWAREA/ BRAILL(3,40),PROOF(40),CODE(40)
      COMMON /OPTS/ EMBOPT,PFOUT
      LOGICAL EMBOPT,PFOUT
C
      CODE(OUTPTR) = X
      IF (.NOT. PFOUT) GO TO 200
      IF (X .EQ. 0) X = 64
      BRAILL(1,OUTPTR) = DOTS14(X)
      BRAILL(2,OUTPTR) = DOTS25(X)
      BRAILL(3,OUTPTR) = DOTS36(X)
      PROOF(OUTPTR) = PCHAR(X)
  200 OUTPTR = OUTPTR + 1
      END


      SUBROUTINE OUTOUT(OUTSIG)
C
C     OUTOUT(OUTSIG) - OUTPUTS CONTENTS OF AND CLEARS OUTPUT BUFFERS.
C
      IMPLICIT INTEGER(A-Z)
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /OPTS/ EMBOPT,PFOUT
```

```
      LOGICAL EMBOPT,PFOUT
      COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),FCHAR(64),MCODE(64)
      COMMON /OWAREA/ BRAILL(3,40),PROOF(40),CODE(40)
      COMMON /MEMB/ EMBOUT(11),MEMBCR,MEMNPG,MIDLE,MEMBON,MEMBOF
      COMMON /CPR/ CPRINT
      LOGICAL CPRINT
      COMMON /LC/ LCOUNT
      COMMON /FILES/ TBDEV,INDEV,PRDEV,BRDEV
    1 FORMAT(80X)
    2 FORMAT(X,40A3)
    3 FORMAT(40(X,I2))
    4 FORMAT(20A4)
C
      IF (.NOT. PFOUT) GO TO 400
      IF (.NOT. CPRINT) WRITE(PRDEV,1)
      CPRINT = .FALSE.
      DO 200 N=1,3
  200 WRITE(PRDEV,2) (BRAILL(N,J),J=1,OUTL)
      WRITE(PRDEV,2) PROOF
      WRITE(PRDEV,3) (CODE(J),J=1,OUTL)
  400 IF (.NOT. EMBOPT) GO TO 1400
      DO 600 N=1,39
      IF (N .GT. OUTL) GO TO 800
      T = CODE(N)
      IF (T .EQ. 0) T = 64
      CALL STCHAR(MCODE(T),EMBOUT,N+2)
  600 CONTINUE
  800 IF (OUTL .GE. 39) GO TO 1200
      DO 1000 N = N,39
 1000 CALL STCHAR(MIDLE,EMBOUT,N+2)
 1200 IF (OUTL .GE. 38) GO TO 1300
      N = OUTL + 1
      CALL STCHAR(MEMBCR,EMBOUT,N+2)
 1300 CONTINUE
      WRITE(BRDEV,4) EMBOUT
 1400 CALL PCLEAR
      IF (OUTSIG .NE. 69) GO TO 1800
      LCOUNT = LPG
      RETURN
 1800 LCOUNT = LCOUNT + 1
```

```
      END
      SUBRØUTINE PAGEHD
C
C     PAGEHD - PAGE HEADING.  IF PAGINATIØN REQUESTED:
C        CØPIES CØNTENTS ØF ØUTPUT BUFFERS TØ TEMPØRARY STØRAGE. SETS UP TITLE
C        IF ANY AND PAGE NUMBER IN ØUTPUT BUFFER. CALLS ØUTØUT. RESTØRES
C        ØRIGINAL ØUTPUT BUFFERS. INCREMENTS PAGE NUMBER.
C
      IMPLICIT INTEGER(A-Z)
      CØMMØN /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCØNT(19),NØELTS(19),PVSTAK(19),NXSTAK(19)
      CØMMØN /ØUT/ ØUTPTR,ØUTL,LINES,LPG
      CØMMØN /ØWAREA/ BRAILL(3,40),PRØØF(40),CØDE(40)
      CØMMØN /TTL/ TTLENG
      CØMMØN /LC/ LCØUNT
      CØMMØN /PAG/ CDIGIT(4),DIGIT(10),PAGINA
      CØMMØN /ØPTS/ EMBØPT,PFØUT
      LØGICAL PFØUT,EMBØPT
      CØMMØN /MEMB/ EMBØUT(11),MEMBCR,MEMBPG,MIDLE,MEMBØN,MEMBØF
      CØMMØN /CPR/ CPRINT
      CØMMØN /DUB/ DØUBLE
      LØGICAL PAGINA,CPRINT,DØUBLE
      CØMMØN /FILES/ TBDEV,INDEV,PRDEV,BRDEV
      DIMENSIØN BTEMP(3,40),CTEMP(40),PTEMP(40)
    1 FØRMAT(80X)
    2 FØRMAT(11A4)
C
      IF (.NØT. PAGINA) RETURN
      IF (PFØUT) WRITE(PRDEV,1)
      IF (.NØT. EMBØPT) GØ TØ 100
      CALL STCHAR(MEMBPG,EMBØUT,3)
      DØ 50 N=4,41
   50 CALL STCHAR(MIDLE,EMBØUT,N)
      WRITE(BRDEV,2) EMBØUT
  100 LCØUNT = 0
      CPRINT = .FALSE.
      N = (ØUTL - TTLENG + 2) / 2 + 1
      IF (N .LT. 1) N = 1
```

```
C     SAVE CONTENTS ØF ØUTPUT BUFFER
      DØ 200 I=1,ØUTL
      PTEMP(I) = PRØØF(I)
      CTEMP(I) = CØDE(I)
      DØ 200 J=1,3
  200 BTEMP(J,I) = BRAILL(J,I)
      CALL PCLEAR
      ØSAVE = ØUTPTR
      ØUTPTR = N
      N = HEADS(2)
  400 IF (NØELTS(N) .EQ. 0 .ØR. SPCØNT(N) .GE. 64) GØ TØ 800
      T = NØELTS(N)
      DØ 600 XYZ = 1,T
  600 CALL MØVEEL(ELT(N,XYZ))
      IF (SPCØNT(N) .EQ. 0) CALL MØVEEL(64)
  800 IF (N .EQ. TAILS(2)) GØ TØ 1000
      N = NXSTAK(N)
      IF (ØUTPTR + NØELTS(N) .LE. ØUTL - 5) GØ TØ 400
 1000 ØUTPTR = ØUTL - 4
      DØ 1200 N=1,4
      IF (CDIGIT(N) .NE. 0) GØ TØ 1400
 1200 ØUTPTR = ØUTPTR + 1
 1400 CALL MØVEEL(60)
      DØ 1600 N=1,4
      XYZ = CDIGIT(N) + 1
 1600 CALL MØVEEL(DIGIT(XYZ))
C
C     INCREMENT PAGE NUMBER
      DØ 1800 N=4,1,-1
      CDIGIT(N) = CDIGIT(N) + 1
      IF (CDIGIT(N) .LT. 10) GØ TØ 2000
 1800 CDIGIT(N) = 0
 2000 CALL ØUTØUT(0)
      IF (DØUBLE) CALL ØUTØUT(0)
C     RESTØRE ØUTPUT BUFFER
      ØUTPTR = ØSAVE
      DØ 2200 I=1,ØUTL
      CØDE(I) = CTEMP(I)
      PRØØF(I) = PTEMP(I)
      DØ 2200 J=1,3
```

```
 2200 BRAILL(J,I) = BTEMP(J,I)
      END


      SUBRØUTINE STAKTB
C
C     STAKTB - SETS STACK PØINTERS TØ TRANSFER A RØW FRØM FREE STACK TØ END ØF
C        CURRENT STACK.
C
      IMPLICIT INTEGER(A-Z)
      CØMMØN /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCØNT(19),NØELTS(19),PVSTAK(19),NXSTAK(19)
C
      IF (HFSTAK .EQ. 0) RETURN
      N = HFSTAK
      HFSTAK = NXSTAK(N)
      NXSTAK(N) = 0
      PVSTAK(N) = TS
      NXSTAK(TS) = N
      TAILS(CURTYP) = N
      TS = N
      CURRS(CURTYP) = N
      CS = N
      IF (STKIND .EQ. 5) STKIND = 2
      END


      SUBRØUTINE PCLEAR
C
C     PCLEAR - RESETS PRØØF ØUTPUT BUFFERS TØ SPACES.
C
      IMPLICIT INTEGER(A-Z)
      CØMMØN /ØWAREA/ BRAILL(3,40),PRØØF(40),CØDE(40)
      CØMMØN /SPAC/ SPACE
C
      DØ 200 I=1,40
      CØDE(I) = 0
      PRØØF(I) = SPACE
      DØ 200 J=1,3
  200 BRAILL(J,I) = SPACE
```

```
      END

      SUBRØUTINE RESET
C
C     RESET - CALLED WHEN $T APPEARS IN INPUT. RESETS PAGE NUMBER, TITLE, AND
C        LØGICAL VARIABLES FØR START ØF NEW DØCUMENT.
C
      IMPLICIT INTEGER(A-Z)
      CØMMØN /STV/ STVAR(10),NSV
      CØMMØN /TTL/ TTLENG
      CØMMØN /N/ LETN
      CØMMØN /INDENT/ MARGIN,INHEAD
      CØMMØN /CØMP/ CØMPUT
      CØMMØN /DUB/ DØUBLE
      LØGICAL DØUBLE,INHEAD,CØMPUT
C
      TTLENG = 0
      MARGIN = 0
      DØUBLE = .FALSE.
      CØMPUT = .FALSE.
      INHEAD = .FALSE.
      DØ 100 I=1,NSV
  100 STVAR(I) = LETN
      END


      SUBRØUTINE DECØDE(N,Y)
C
C     DECØDE - DECØDES CHARACTER REPRESENTATIØN ØF INTEGER. USED IN SKIP-LINES
C        AND SETTING TABS AND MARGIN.
C
      IMPLICIT INTEGER(A-Z)
      CØMMØN /BUFF/ BUFFER(10)
C
      Y = 0
      DØ 200 I=1,N
  200 Y = 10 * Y + IAND(15,ICH(BUFFER,4*I-3))
      END
```

```
C
C                          RJE TAPE FILE HANDLING PROGRAM
C
C     PERFORMS ONE OF THE FOLLOWING FUNCTIONS DEPENDING ON THE PROGRAM NUMBER
C     (INTEGER ON THE FIRST CARD OF THE INPUT)
C     1  TAKES CONTINUOUS INPUT FROM DEVICE F:105 (CARD READER) AND PRODUCES A
C        LINE BY LINE RJE FILE IMAGE ON DEVICE F:119 (DISC FILE) FROM WHICH AN
C        RJE FILE MAY BE CREATED USING !RJBUILD
C     2  LISTS RJE FILES ON DEVICE F:108 (LINEPRINTER)
C     3  CONVERTS RJE FILES INTO A SUITABLE FORMAT FOR INPUT TO DOTSYS III,
C        OUTPUT ON DEVICE F:121 (TAPE OR DISC FILE)
C     4  COPIES RJE FILES TO ARCHIVE TAPE
C     5  RJE CONVERSION FOLLOWED BY DOTSYS (FORTRAN IMPROVED VERSION)
C     6  COPIES RJE FILE "BRAILLE" TO FGDTEMP FOR EMBOSSING
C     7  DOTSYS (FORTRAN IMPROVED VERSION) WITHOUT RJE CONVERSION
C
      IMPLICIT INTEGER(A-Z)
      COMMON /SUB/ SUBNO,VERSNO
      COMMON /INDEX/ INDEX(62)
    1 FORMAT(2I)
    2 FORMAT(' ERROR IN PROGRAM NUMBER')
    4 FORMAT(' TYPE CORRECT NUMBER')
      READ(104,1) SUBNO,VERSNO
      IF (SUBNO .GT. 1 .AND. SUBNO .LT. 7)
     *   CALL BUFFERIN(120,1,INDEX,62,ISTAT)
      IF (SUBNO .LE. 0 .OR. SUBNO .GE. 8) GO TO 200
      IF (SUBNO .EQ. 7) GO TO 160
  150 CALL SEGLOD(1)
      IF (SUBNO .EQ. 1) CALL CARDS
      IF (SUBNO .EQ. 2) CALL LIST
      IF (SUBNO .EQ. 3) CALL RJECON
      IF (SUBNO .EQ. 4) CALL SAVE
      IF (SUBNO .NE. 5) GO TO 170
      CALL RJECON
  160 CALL SEGLOD(2)
      CALL DOTSYS
      GO TO 180
  170 IF (SUBNO .EQ. 6) CALL EMBOSS
  180 CONTINUE
      STOP 'OK'
```

```
  200 WRITE(115,2)
      WRITE(115,4)
S        CAL1,9    9          WAIT
      READ(115,1) SUBNO,VERSNO
      IF (SUBNO .GE. 1 .AND. SUBNO .LE. 7) GO TO 150
      STOP 'ERROR'
      END


      SUBROUTINE CARDS
C     RJE FILE FROM CARDS.  MACRO FACILITY AVOIDS REPETITIOUS TYPING:
C     MACRO DEFINITION FORMAT:  @@<CHARACTER-PAIR>=<TEXT>@
C     MACRO CALL FORMAT:  @<CHARACTER-PAIR>
      IMPLICIT INTEGER(A-Z)
      COMMON /SUB/ SUBNO,VERSNO
      COMMON /RDCH/ ENDRED,INTEMP(20),INPUT(20),INPTR,CHAR
     *   ,INLENG,SPACE
      COMMON /EB/ EBCEXT(256)
      DIMENSION EBCDIC(255)
      EGUIVALENCE(EBCDIC,EBCEXT(2))
      DIMENSION MACTAB(1000),MACSIZ(100),MCBASE(100),MCCHAR(100)
      LOGICAL INMAC,ENDRED,WARN
      DIMENSION OUTPUT(80)
      DATA MTSIZ,NMMAX /4000,100/
      DATA MACSYM,EQUALS/2Z7C,2Z7E/
      DATA SPACE,LSPACE,H1,H2/2Z40,8Z40000000,2Z80,2ZC0/
      DATA INLENG,OLENG/80,72/
    1 FORMAT(80A1)
    2 FORMAT(X,80A1)
   10 FORMAT(' SEE LP')
   11 FORMAT(20X,'MACRO REDEFINED',A4)
   12 FORMAT(' TOO MANY MACROS')
   13 FORMAT(' MISSING ''='' SIGN')
   14 FORMAT(' MACRO TABLE FULL')
   15 FORMAT(20X,'UNDEFINED MACRO IGNORED',A4)
C     OLENG = MAXIMUM LENGTH OF OUTPUT LINE (< 80 CHARACTERS)
      INMAC = .FALSE.
      ENDRED = .FALSE.
      WARN = .FALSE.
      MTPTR = 0
```

```
      LAST = SPACE
      INPTR = INLENG
      ØPTR = 0
      LINEND = ØLENG
      DØ 100 I=1,256
  100 EBCEXT(I) = I - 1
      IF (VERSN .NE. 1) GØ TØ 140
C
C     VERSIØN NUMBER 1 INDICATES CØDE CØNVERSIØN SPECIFIED
  110 CALL BUFFERIN(104,1,INTEMP,20,ISTAT)
      DØ 120 I=1,20
      IF (ICH(INTEMP,4*I-2) .NE. EQUALS .ØR. ICH(INTEMP,4*I) .NE. SPACE)
     *    GØ TØ 140
  120 EBCDIC(ICH(INTEMP,4*I-3)) = ICH(INTEMP,4*I-1)
      GØ TØ 110
  140 CALL BUFFIN(105,1,INTEMP,20)
C
  150 IF (ENDRED) GØ TØ 500
      IF (INMAC) GØ TØ 860
      CALL RDCHAR
      IF (ENDRED) CHAR = SPACE
      IF (CHAR .EQ. SPACE .AND. LAST .EQ. SPACE) GØ TØ 150
      IF (CHAR .EQ. MACSYM) GØ TØ 1000
  270 ØPTR = ØPTR + 1
      ØUTPUT(ØPTR) = CHAR
      LAST = CHAR
      IF (CHAR .EQ. SPACE) LINEND = ØPTR - 1
      IF (ØPTR .GT. ØLENG) GØ TØ 300
      GØ TØ 150
  300 REST = 80 - LINEND
      DØ 350 I=1,LINEND
  350 ØUTPUT(I) = ISL(ØUTPUT(I),24)
      WRITE(119,1) (ØUTPUT(I),I=1,LINEND),(LSPACE,I=1,REST)
      MØVENØ = ØLENG - LINEND
      IF (MØVENØ .EQ. 0) GØ TØ 450
      DØ 400 I=1,MØVENØ
  400 ØUTPUT(I) = ØUTPUT(LINEND+I+1)
  450 ØPTR = MØVENØ
      GØ TØ 150
```

```
  500 REST = 80 - LINEND
      DØ 550 I=1,LINEND
  550 ØUTPUT(I) = ISL(ØUTPUT(I),24)
      WRITE(119,1) (ØUTPUT(I),I=1,LINEND),(LSPACE,I=1,REST)
      ENDFILE 119
      IF (WARN) WRITE(115,10)
      STØP 'ØK'
  700 STØP 'MACRØ ERRØR'
C
  860 CØUNT = CØUNT + 1
      IF (CØUNT .LE. MCSIZ) GØ TØ 880
      INMAC = .FALSE.
      GØ TØ 150
  880 CHAR = ICH(MACTAB,MCBAS+CØUNT)
      GØ TØ 270
C
 1000 CALL RDCHAR
      IF (CHAR .EQ. MACSYM) GØ TØ 2000
C
C     MACRØ CALL
      T = CHAR
      CALL RDCHAR
      CHAR = 256 * T + CHAR
      DØ 1100 I=1,NMACS
 1100 IF (CHAR .EQ. MCCHAR(I)) GØ TØ 1200
      WRITE(108,15) CHAR
      WARN = .TRUE.
      GØ TØ 150
 1200 MCSIZ = MACSIZ(I)
      MCBAS = MCBASE(I)
      CØUNT = 0
      INMAC = .TRUE.
      GØ TØ 150
C
C     MACRØ DEFINITIØN
 2000 CALL RDCHAR
      IF (NMACS .EQ. 0) GØ TØ 2200
      DØ 2100 I=1,NMACS
      IF (CHAR .NE. MCCHAR(I)) GØ TØ 2100
```

```
      WRITE(108,11) MCCHAR(I)
      MCCHAR(I) = 0
      WARN = .TRUE.
      GØ TØ 2200
 2100 CØNTINUE
 2200 NMACS = NMACS + 1
      IF (NMACS .LE. NMMAX) GØ TØ 2300
      WRITE(115,12)
      GØ TØ 700
 2300 T = CHAR
      CALL RDCHAR
      MCCHAR(NMACS) = 256 * T + CHAR
      CALL RDCHAR
      IF (CHAR .EQ. EQUALS) GØ TØ 2400
      WRITE(115,13)
      GØ TØ 700
 2400 MCBASE(NMACS) = MTPTR
 2500 CALL RDCHAR
      IF (CHAR .EQ. MACSYM) GØ TØ 2700
      MTPTR = MTPTR + 1
      IF (MTPTR .LE. MTSIZ) GØ TØ 2600
      WRITE(115,14)
      GØ TØ 700
 2600 CØNTINUE
C     ICH(MACTAB,MTPTR) = CHAR
S        LW,7        MTPTR
S        AI,7        -1
S        LW,9        CHAR
S        STB,9       MACTAB,7
      GØ TØ 2500
 2700 MACSIZ(NMACS) = MTPTR - MCBASE(NMACS)
      GØ TØ 150
      END


      SUBRØUTINE RDCHAR
C     READS NEXT CHARACTER FRØM INPUT  (CALLED BY CARDS)
      IMPLICIT INTEGER(A-Z)
      CØMMØN /RDCH/ ENDRED,INTEMP(20),INPUT(20),INPTR,CHAR
     *   ,INLENG,SPACE
```

```
      COMMON /EB/ EBCEXT(256)
      DIMENSION EBCDIC(255)
      EQUIVALENCE(EBCDIC,EBCEXT(2))
      LOGICAL ENDRED
      DATA H1,H2/2Z80,2ZC0/
    1 FORMAT(' ILLEGAL INPUT CHARACTER')
      INPTR = INPTR + 1
      IF (INPTR .LE. INLENG) GO TO 200
   80 IF (ICHECK(105) .EQ. 1) GO TO 80
      IF (ICHECK(105) .NE. 3) GO TO 100
      ENDRED = .TRUE.
      RETURN
  100 DO 120 I=1,20
  120 INPUT(I) = INTEMP(I)
      CALL BUFFIN(105,1,INTEMP,20)
      INPTR = 1
  200 CHAR = EBCDIC(ICH(INPUT,INPTR))
      IF (CHAR.LT.SPACE .OR. CHAR.GE.H1 .AND. CHAR.LE.H2) GO TO 600
      RETURN
  600 WRITE(115,1)
S          CAL1,9   9               WAIT
      END


      SUBROUTINE LIST
C    LISTS ANY NUMBER OF FILES IN THE ORDER IN WHICH THEY OCCUR ON THE TAPE
      IMPLICIT INTEGER(A-Z)
      EXTERNAL FLIST
      CALL RDLOC
      CALL FCOPY(FLIST)
      END


      SUBROUTINE RJECON
C    CONVERTS ANY NUMBER OF FILES IN THE ORDER IN WHICH THEY OCCUR ON THE TAPE
      IMPLICIT INTEGER(A-Z)
      DIMENSION TABLES(2),ENDLIN(20)
      COMMON /BUFF/ BUFFER(20)
      COMMON /NAMES/ NAMTAB(20,2),FPOSN(20),NAME(2),NFILES
      COMMON /INDEX/ INDEX(62)
```

```
      COMMON /F/ FILNO
      COMMON /C/ NLINES,LOGTAB
      COMMON /SUB/ SUBNO,VERSNO
      EQUIVALENCE (NFOT,INDEX(4))
      EXTERNAL CCOPY
      DATA TABLES/'TABL','ES8('/
      DATA SPACE,SPACES,OLD/2Z40,'        ','OLD '/
      DATA ENDLIN / '$T   ',19*'    ' /
    1 FORMAT(' RJE TAPE FULL')
    2 FORMAT(//X,I,'CARD IMAGES WRITTEN'/)
      NLINES = 0
      FILNO = -1
      OUTPUT 'RJE CONVERSION'
      CALL DATE
      IF (SUBNO .NE. 5 .OR. NFOT .LT. 19) GO TO 500
      WRITE(115,1)
S         CAL1,9    3            ABORT
  500 IF (SUBNO .NE. 5) CALL BUFFERIN(105,1,NEWTAB,1,ISTAT)
      CALL RDLOC
      IF (SUBNO .EQ. 5 .OR. NEWTAB .NE. OLD) GO TO 630
  620 CALL BUFFERIN(121,1,BUFFER,20,ISTAT)
      IF (BUFFER(20) .NE. SPACES) GO TO 620
      GO TO 700
C     LOCATE AND COPY TABLES
  630 CALL LOCATE(TABLES,TABPOS)
      IF (TABPOS .EQ. 0) STOP 'TABLES NOT ON TAPE'
      CALL SKIPFILE(120,TABPOS)
      DO 650 I=1,2
  650 NAME(I) = TABLES(I)
      CALL CCOPY
      IF (SUBNO .EQ. 5) CALL COPBAK(NFOT-TABPOS+1,.TRUE.)
  700 REWIND 120
      FILNO = 0
      CALL FCOPY(CCOPY)
      CALL BUFFEROUT(121,1,ENDLIN,20,ISTAT)
      IF (ISTAT .NE. 2) STOP 'OUTPUT ERROR'
      IF (SUBNO .EQ. 5) CALL COPBAK(1,.FALSE.)
      ENDFILE 121
      WRITE(108,2) NLINES
      END
```

```
      SUBRØUTINE CØPBAK(SKIPNØ,LØGTAB)
C     CØPIES DISC FILE BACK TØ TAPE  (CALLED BY RJECØN)
      IMPLICIT INTEGER(A-Z)
      CØMMØN /BUFF/ BUFFER(20)
      LØGICAL LØGTAB
      DATA SPACES / '    ' /
      ENDFILE 121
      REWIND 121
      CALL SKIPFILE(120,SKIPNØ)
      IF (LØGTAB) GØ TØ 200
  100 CALL BUFFERIN(120,1,BUFFER,20,ISTAT)
      IF (BUFFER(20) .NE. SPACES) GØ TØ 100
  200 CALL BUFFERIN(121,1,BUFFER,20,ISTAT)
      IF (ISTAT .EQ. 3) GØ TØ 400
      CALL BUFFERØUT(120,1,BUFFER,20,ISTAT)
      GØ TØ 200
  400 ENDFILE 120
      REWIND 121
      IF (.NØT. LØGTAB) CALL SKIPFILE(120,-1)
      END


      SUBRØUTINE RDLØC
C     READ FILE NAMES AND LØCATE IN INDEX
      IMPLICIT INTEGER(A-Z)
      CØMMØN /NAMES/ NAMTAB(20,2),FPØSN(20),NAME(2),NFILES
    1 FØRMAT(X,2A4,' NØT ØN TAPE')
    4 FØRMAT(2A4)
      DØ 200 I=1,20
  200 FPØSN(I) = 0
      DØ 300 I=1,20
  300 READ(105,4,END=400) (NAMTAB(I,J),J=1,2)
  400 NFILES = I - 1
      IF (NFILES .EQ. 0) RETURN
      DØ 600 I=1,NFILES
      DØ 500 J=1,2
  500 NAME(J) = NAMTAB(I,J)
      CALL LØCATE(NAME,PØSN)
      IF (PØSN .NE. 0) GØ TØ 550
      WRITE(115,1) NAME
```

```
S          CAL1,9    9            WAIT
      GØ TØ 600
  550 FPØSN(PØSN) = I
  600 CØNTINUE
      END


      SUBRØUTINE FCØPY(CØPY)
C     CØPY FILES IN APPRØPRIATE FØRMAT
C     LEAVE TAPE PØSITIØNED BEFØRE LAST END ØF FILE MARK
      IMPLICIT INTEGER(A-Z)
      CØMMØN /NAMES/ NAMTAB(20,2),FPØSN(20),NAME(2),NFILES
      CØMMØN /INDEX/ INDEX(62)
      EGUIVALENCE (INDEX(4),NFØT)
      DØ 1000 I=1,NFØT
      CALL SKIPFILE(120,1)
      IF (NFILES .EQ. 0 .ØR. FPØSN(I) .EQ. 0) GØ TØ 1000
      DØ 900 K=1,2
  900 NAME(K) = NAMTAB(FPØSN(I),K)
      CALL CØPY
 1000 CØNTINUE
      END


      SUBRØUTINE FLIST
C     LISTS RJE FILE ØN LP
      IMPLICIT INTEGER(A-Z)
      CØMMØN /NAMES/ NAMTAB(20,2),FPØSN(20),NAME(2),NFILES
      DIMENSIØN INPUT(90),ØUTPUT(26)
      LØGICAL CØNTRL
      DATA DQUØT,G,DØLLAR,SQUØT/2Z7F,2ZC7,2Z5B,2Z7D/
      DATA LDQUØT,LSQUØT,LGRAD /' "  ',' $'' ',' $G '/
      DATA SPACES /'    '/
      DATA LSTAR /'*'/
    1 FØRMAT('1WARWICK RESEARCH UNIT FØR THE BLIND REMØTE JØB ENTRY TAPE
     *')
    2 FØRMAT(2X,I5,5X,80A1)
    3 FØRMAT(' LISTING ØF FILE NAMED ',2A4/)
    4 FØRMAT(/' *****',A4,'ØCCURS ',I,'TIMES')
    5 FØRMAT(95X,'WARNING: ',I,'CHARACTERS')
```

```
      CØNTRL = .FALSE.
      NGRADS = 0
      NSQUØT = 0
      NDQUØT = 0
      DØ 100 I=1,3
  100 ØUTPUT(I) = SPACES
      INPTR = 1
      WRITE(108,1)
      CALL DATE
      WRITE(108,3) NAME
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
  230 NBYTES = ICH(INPUT,INPTR)
      IF (NBYTES .EQ. 0) GØ TØ 300
      DØ 235 I=3,26
  235 ØUTPUT(I) = SPACES
      IF (NBYTES .LE. 80) GØ TØ 240
      WRITE(108,5) NBYTES
      IF (NBYTES .GT. 90) STØP 'NEXT LINE TØØ LØNG'
C
  240 DØ 250 I=1,NBYTES
      IF (INPTR + I .LE. 360) GØ TØ 242
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
      INPTR = INPTR - 360
  242 CHAR = ICH(INPUT,INPTR+I)
      IF (.NØT. CØNTRL) GØ TØ 245
C     CØUNT $G AND QUØTES, ØUTPUT * ØN LINES WHERE THEY ØCCUR
      IF (CHAR .NE. G) GØ TØ 243
      NGRADS = NGRADS + 1
      ØUTPUT(3) = LSTAR
      GØ TØ 247
  243 IF (CHAR .NE. SQUØT) GØ TØ 245
      NSQUØT = NSQUØT + 1
      ØUTPUT(3) = LSTAR
      GØ TØ 247
  245 IF (CHAR .NE. DQUØT) GØ TØ 247
      NDQUØT = NDQUØT + 1
      ØUTPUT(3) = LSTAR
  247 CØNTRL = CHAR .EQ. DØLLAR
C     ICH(ØUTPUT,12+I) = ISL(CHAR,24)
S         LW,7       I
```

```
S         AI,7      11
S         LW,9      CHAR
S         STB,9     ØUTPUT,7
  250 CØNTINUE
C
C     INCREMENT LINE NUMBER
S         LI,7      7
S256      LB,9      ØUTPUT,7
S         CI,9      X'40'
S         BNE       $+2
S         LI,9      X'F0'
S         AI,9      1
S         CI,9      X'FA'
S         BNE       260S
S         LI,9      X'F0'
S         STB,9     ØUTPUT,7
S         BDR,7     256S
S260      STB,9     ØUTPUT,7
      CALL BUFFERØUT(108,1,ØUTPUT,26,ISTAT)
  270 INPTR = NBYTES + INPTR + 1
      IF (INPTR .LE. 360) GØ TØ 230
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
      INPTR = INPTR - 360
      GØ TØ 230
  300 IF (NSQUØT/2*2 .NE. NSQUØT) WRITE(108,4) LSQUØT,NSQUØT
      IF (NDQUØT/2*2 .NE. NDQUØT) WRITE(108,4) LDQUØT,NDQUØT
      IF (NGRADS/2*2 .NE. NGRADS) WRITE(108,4) LGRAD,NGRADS
      END


      SUBRØUTINE LØCATE(FNAME,FPØS)
C     FINDS PØSN ØF FILE ØN TAPE, RETURNS 0 IF ABSENT
      IMPLICIT INTEGER(A-Z)
      CØMMØN /INDEX/ INDEX(62)
      EQUIVALENCE (NFØT,INDEX(4))
      DIMENSIØN FNAME(2)
      DATA SPACE/2Z40/
      DATA NEWLIN/1ZD/
      DØ 200 I=1,NFØT
      BASE = 8 * (I + 1)
```

```
      DØ 100 J=1,8
      CHAR = ICH(INDEX,BASE+J)
      CHAR1 = ICH(FNAME,J)
      IF (CHAR .EQ. NEWLIN .AND. CHAR1 .EQ. SPACE) GØ TØ 300
  100 IF (CHAR .NE. CHAR1) GØ TØ 200
      GØ TØ 300
  200 CØNTINUE
      FPØS = 0
      RETURN
  300 FPØS = I
      END


      SUBRØUTINE CCØPY
C     CØNVERTS FILE TØ CARD IMAGE FØRMAT FØR INPUT TØ DØTSYS
C     LØGTAB IS TRUE FØR TABLES, ØTHERWISE FALSE
      IMPLICIT INTEGER(A-Z)
      CØMMØN /LWR/ ØUTPUT(20),ØPBASE,LENGTH,SPACE
      CØMMØN /C/ NLINES,LØGTAB
      CØMMØN /F/ FILNØ
      CØMMØN /NAMES/ NAMTAB(20,2),FPØSN(20),NAME(2),NFILES
      DIMENSIØN INPUT(90)
      LØGICAL LØGTAB,HYPHEN
      DATA PAGE,STAR/'$PG ',8Z5C000000/
      DATA SPACE /'    '/
      DATA RSPACE / 2Z40 /
      DATA HYPH,SEMIC,ØSB,CSB,LT,GT/2Z60,2Z7A,2ZB4,2ZB5,2Z4C,2Z6E/
    1 FØRMAT('   $PG $%$%$%$% $%$%$%$% $SL01 ',I3,X,2A4,'$SL01 $%$%$%$%
     *$%$%$%$% $T',11X)
    2 FØRMAT(X,I4,4X,2A4,8X,20A4)
    3 FØRMAT(' TABLE ERRØR')
    4 FØRMAT(' MISSING * IN ',2A4,3X,'FIRST LINE IS'/X,20A4)
      CHAR = 0
      DØ 50 I=1,20
   50 ØUTPUT(I) = SPACE
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
      LØGTAB = FILNØ .LT. 0
      FILNØ = FILNØ + 1
      IF (LØGTAB) GØ TØ 120
      WRITE(121,1) FILNØ,NAME
```

```
      LENGTH = 80
      NBYTES = ICH(INPUT,1)
      IF (NBYTES .GT. 80) STOP 'LINE TOO LONG'
      DO 100 I=1,NBYTES
C     ICH(OUTPUT,I) = ISL(ICH(INPUT,I+1),24)
S        LW,7       I
S        LB,9       INPUT,7
S        AI,7       -1
S        STB,9      OUTPUT,7
  100 CONTINUE
S        LB,9       OUTPUT
S        CB,9       STAR
S        BE         103S
      WRITE(115,4) NAME,OUTPUT
S          CAL1,9   9             WAIT
  103 WRITE(108,2) FILNO,NAME,OUTPUT
      OUTPUT(1) = PAGE
      INPTR = NBYTES + 2
      OPBASE = 4
      GO TO 150
  120 LENGTH = 80
      OPBASE = 0
      INPTR = 1
  150 NBYTES = ICH(INPUT,INPTR)
      IF (NBYTES .EQ. 0) GO TO 500
      IF (.NOT. LOGTAB .OR. NBYTES .EQ. 80) GO TO 155
      WRITE(115,3)
S          CAL1,9   3             ABORT
  155 HYPHEN = .FALSE.
C
      DO 300 I=1,NBYTES
      IF (INPTR + I .LE. 360) GO TO 160
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
      INPTR = INPTR - 360
  160 OUTPTR = OPBASE + I
C     CHAR = ICH(INPUT,INPTR+I)
S        LW,7       INPTR
S        AW,7       I
S        AI,7       -1
S        LB,9       INPUT,7
```

```
S        STW,9     CHAR
      IF (LOGTAB) GO TO 170
      IF (CHAR .EQ. OSB) CHAR = LT
      IF (CHAR .EQ. CSB) CHAR = GT
C     REPLACE LOWER CASE LETTERS WITH UPPER CASE
      IF (CHAR .GT. 2Z80 .AND. CHAR .LT. 2ZC0) CHAR = CHAR + 2Z40
      HYPHEN = CHAR .EQ. HYPH .AND. OLAST .NE. SEMIC .AND.
     *   OLAST .NE. RSPACE
      OLAST = CHAR
  170 CONTINUE
C     ICH(OUTPUT,OUTPTR) = ISL(CHAR,24)
S        LW,7      OUTPTR
S        AI,7      -1
S        LW,9      CHAR
S        STB,9     OUTPUT,7
      IF (I .EQ. NBYTES .AND. HYPHEN) OUTPTR = OUTPTR - 1
      IF (OUTPTR .GE. LENGTH) CALL LWRITE
  300 CONTINUE
C
      OPBASE = OUTPTR
      IF (OPBASE .EQ. LENGTH) OPBASE = 0
      IF (HYPHEN .OR. LOGTAB) GO TO 400
      OPBASE = OPBASE + 1
      IF (OPBASE .GE. LENGTH) CALL LWRITE
  400 INPTR = INPTR + NBYTES + 1
      IF (INPTR .LE. 360) GO TO 150
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
      INPTR = INPTR - 360
      GO TO 150
  500 IF (.NOT. LOGTAB) CALL LWRITE
      END


      SUBROUTINE LWRITE
C     OUTPUTS A LINE IN CARD IMAGE FORMAT  (CALLED BY CCOPY)
      IMPLICIT INTEGER(A-Z)
      COMMON /LWR/ OUTPUT(20),OPBASE,LENGTH,SPACE
      COMMON /C/ NLINES,LOGTAB
      LOGICAL LOGTAB
    1 FORMAT(' TOO MUCH INPUT')
```

```
            CALL BUFFEROUT(121,1,OUTPUT,20,ISTAT)
            IF (ISTAT .EQ. 2) GO TO 200
            WRITE(115,1)
S              CAL1,9    3            ABORT
       200  OPBASE = OPBASE - LENGTH
C           DO I=1,20 OUTPUT(I) = SPACE
S              LW,9       SPACE
S              LI,7       20
S              STW,9      OUTPUT-1,7
S              BDR,7      $-1
            IF (.NOT. LOGTAB) NLINES = NLINES + 1
            END


            SUBROUTINE EMBOSS
C           COPIES RJE FILE NAMED BRAILLE TO FGDTEMP FOR EMBOSSING
            IMPLICIT INTEGER(A-Z)
            DIMENSION INPUT(90),OUTPUT(11),FNAME(2)
            LOGICAL FIRST,ENDRED,INHEAD,ENDWRT
            DATA FNAME / 'BRAI','LLE ' /
            DATA OP1,OUTPUT(11),SPACES / 8Z0A124040,8Z0A3C0000,'    ' /
            DATA NEWPAG,NEWDOC / 8Z0A120C0A,8Z0A127E7E /
          1 FORMAT(' BRAILLE NOT ON TAPE')
          2 FORMAT(' BRAILLE ERROR')
          3 FORMAT(/' DOCUMENT NO.   START   END   #PAGES   #LINES   TIME'/)
          4 FORMAT(' EOF ON FGDTEMP AT PAGE ',I)
          5 FORMAT(6X,I2,8X,I4,4X,I3,5X,I3,5X,I4,5X,I3)
          6 FORMAT(/X,I,'PAGES'/X,I,'LINES'/' ESTIMATED EMBOSSING TIME ',I,'MI
         *NUTES'/)
         50 CALL LOCATE(FNAME,FPOS)
            IF (FPOS .NE. 0) GO TO 100
            WRITE(115,1)
S              CAL1,9    9            WAIT
            REWIND 120
            GO TO 50
        100 CALL SKIPFILE(120,FPOS)
            ENDRED = .FALSE.
            ENDWRT = .FALSE.
            INHEAD = .FALSE.
            FIRST = .TRUE.
```

```
      LINES = 0
      PAGES = 0
      TØTLIN = 0
      TØTPAG = 0
      TØTTIM = 0
      DØCNØ = 0
      INPTR = 1
      WRITE(108,3)
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
  230 ØUTPUT(1) = ØP1
      DØ 232 I=2,10
  232 ØUTPUT(I) = SPACES
      NBYTES = ICH(INPUT,INPTR)
      IF (NBYTES .NE. 0) GØ TØ 234
      ENDRED = .TRUE.
      GØ TØ 400
  234 IF (NBYTES .LE. 40) GØ TØ 235
      WRITE(115,2)
      STØP 'ERRØR'
  235 DØ 250 I=3,NBYTES
      IF (INPTR + I .LE. 360) GØ TØ 240
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
      INPTR = INPTR - 360
  240 CØNTINUE
C     ICH(ØUTPUT,I) = ICH(INPUT,INPTR+I)
S        LW,7       INPTR
S        AW,7       I
S        AI,7       -1
S        LB,9       INPUT,7
S        SW,7       INPTR
S        STB,9      ØUTPUT,7
  250 CØNTINUE
      IF (ØUTPUT(1) .EQ. NEWDØC) GØ TØ 300
      IF (ØUTPUT(1) .EQ. NEWPAG) GØ TØ 255
      LINES = LINES + 1
      GØ TØ 260
  255 PAGES = PAGES + 1
  260 IF (ENDWRT) GØ TØ 270
      CALL BUFFERØUT(118,1,ØUTPUT,11,ISTAT)
      IF (ISTAT .EQ. 2) GØ TØ 270
```

```
      T = TØTPAG + PAGES + DØCNØ
      WRITE(115,4) T
      ENDWRT = .TRUE.
  270 INPTR = NBYTES + INPTR + 1
      IF (INPTR .LE. 360) GØ TØ 230
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
      INPTR = INPTR - 360
      GØ TØ 230
C
  300 IF (.NØT. INHEAD) GØ TØ 350
      INHEAD = .FALSE.
      GØ TØ 260
  350 IF (.NØT. FIRST) GØ TØ 400
      FIRST = .FALSE.
      GØ TØ 500
  400 DØCNØ = DØCNØ + 1
      START = TØTPAG + DØCNØ
      TØTLIN = TØTLIN + LINES
      IF (.NØT. ENDRED) PAGES = PAGES - 1
      TØTPAG = TØTPAG + PAGES
      FINISH = TØTPAG + DØCNØ
      TIME = (PAGES + LINES + 7) / 14
      TØTTIM = TØTTIM + TIME
      WRITE(108,5) DØCNØ,START,FINISH,PAGES,LINES,TIME
      IF (ENDRED) GØ TØ 1000
  500 INHEAD = .TRUE.
      LINES = 0
      PAGES = 0
      GØ TØ 260
 1000 TØTPAG = TØTPAG + DØCNØ
      WRITE(108,6) TØTPAG,TØTLIN,TØTTIM
      ENDFILE 118
      STØP 'ØK'
      END


      SUBRØUTINE DATE
C     PRINTS CURRENT DATE ØN LP
      IMPLICIT INTEGER(A-Z)
      DIMENSIØN DT(4)
```

```
      1 FØRMAT(/' DATE:
S           LI,9        DT
S           AW,9        =8Z10000000
S           CAL1,8      9
        DT(1) = ISL(DT(3),16)
        DT(2) = ISL(DT(2),16) + ISL(ICH(DT(3),1),8)
        DT(3) = ISL(DT(4),16)
        IF (ICH(DT(1),1) .EQ. 2ZF0) DT(1) = DT(1) - 8ZB0000000
        WRITE(108,1) (DT(I),I=1,3)
        END


        SUBRØUTINE SAVE
C       CØPIES SPECIFIED FILES FRØM RJE TAPE TØ ARCHIVE TAPE VIA DISC
C       IF NØ FILE NAMES SPECIFIED, ALL FILES WITH * AS FIRST CHAR ARE CØPIED
        IMPLICIT INTEGER(A-Z)
        CØMMØN /MØV/ BUFFER(1800),BUFPTR,BUFSIZ,INPUT(90)
        CØMMØN /INDEX/ INDEX(62)
        CØMMØN /NAMES/ NAMTAB(20,2),FPØSN(20),NAME(2),NFILES
        EQUIVALENCE (NFØT,INDEX(4)),(NRECS,INPUT(4))
        DIMENSIØN SINDEX(4,19),TIME(4)
        DATA BUFSIZ,STAR /1800,2Z5C/
      1 FØRMAT(X,2A4,' NØT ARCHIVED - MISSING ASTERISK IN FIRST LINE')
      2 FØRMAT(X,2A4,' NØT ARCHIVED - FILE ØF SAME NAME ALREADY ØN TAPE')
      3 FØRMAT(X,2A4,' ARCHIVED')
        CALL RDLØC
        IF (NFILES .NE. 0) GØ TØ 100
        DØ 80 I=1,NFØT
     80 FPØSN(I) = 1
    100 NSAVED = 0
C       FØLLØWING CØDE PLACES DATE AND TIME IN TIME
S           LI,9        TIME
S           AW,9        =8Z10000000
S           CAL1,8      9
        DØ 1000 I=1,NFØT
        CALL SKIPFILE(120,1)
        IF (FPØSN(I) .EQ. 0) GØ TØ 1000
S           LI,7        84
S           AW,7        I
S           LH,9        INDEX,7
```

```
S        STW,9     NLIN
S        AI,7      20
S        LH,9      INDEX,7
S        STW,9     NREC
      IF (NREC .EQ. 0) GO TO 1000
      DO 200 J=1,NREC
      CALL BUFFERIN(120,1,INPUT,90,ISTAT)
      IF (J .NE. 1) GO TO 150
      IF (ICH(INPUT,2) .EQ. STAR) GO TO 150
      WRITE(108,1) (INDEX(2*I+J),J=3,4)
      GO TO 1000
  150 CALL BUFFEROUT(119,1,INPUT,90,ISTAT)
  200 IF (ISTAT .EQ. 3) GO TO 1200
      NSAVED = NSAVED + 1
      SINDEX(1,NSAVED) = INDEX(2*I+3)
      SINDEX(2,NSAVED) = INDEX(2*I+4)
      SINDEX(3,NSAVED) = NLIN
      SINDEX(4,NSAVED) = NREC
 1000 CONTINUE
      GO TO 1400
 1200 OUTPUT(115) 'EXCEEDED DISC SPACE'
 1400 REWIND 120
      REWIND 119
      IF (NSAVED .EQ. 0) RETURN
      DO 1500 I=1,NSAVED
 1500 FPOSN(I) = 1
      OUTPUT(115) 'LOAD ARCHIVE TAPE ON TC'
S        CAL1,9    9           BEGIN WAIT
      CALL SKIPFILE(120,1)
      BUFPTR = BUFSIZ
      CALL IMOVE(0)
 1600 CALL IMOVE(4)
      IF (INPUT(1) .EQ. 0) GO TO 1780
      DO 1700 I=1,NSAVED
 1700 IF (SINDEX(1,I) .EQ. INPUT(1) .AND. SINDEX(2,I) .EQ. INPUT(2))
     *   GO TO 1720
      GO TO 1750
 1720 WRITE(108,2) (INPUT(J),J=1,2)
      FPOSN(I) = 0
 1750 DO 1770 I=1,NRECS
```

```
 1770 CALL I
      GØ TØ 1600
 1780 BUFPTR = BUFPTR - 90
      BACKSPACE 120
      DØ 2400 I=1,NSAVED
      DØ 1800 J=1,4
 1800 INPUT(J) = SINDEX(J,I)
      IF (FPØSN(I) .EQ. 0) GØ TØ 2300
      INPUT(5) = ISL(TIME(3),16)
      INPUT(6) = ISL(TIME(2),16) + ISL(ICH(TIME(3),1),8)
      INPUT(7) = ISL(TIME(4),16)
      DØ 2000 J=8,90
 2000 INPUT(J) = 0
      CALL ØMØVE
      DØ 2200 J=1,NRECS
      CALL BUFFERIN(119,1,INPUT,90,ISTAT)
 2200 CALL ØMØVE
      GØ TØ 2400
 2300 DØ 2350 J=1,NRECS
 2350 CALL BUFFERIN(119,1,INPUT,90,ISTAT)
 2400 CØNTINUE
      DØ 2600 I=1,90
 2600 INPUT(I) = 0
      CALL ØMØVE
      CALL BUFFERØUT(120,1,BUFFER,BUFSIZ,ISTAT)
      ENDFILE 120
      ENDFILE 120
      DØ 2800 I=1,NSAVED
 2800 IF (FPØSN(I) .NE. 0) WRITE(108,3) (SINDEX(J,I),J=1,2)
      END


      SUBRØUTINE IMØVE(LENG)
C     GETS FIRST 4 WØRDS ØF NEXT LØGICAL RECØRD FRØM ARCHIVE TAPE
      IMPLICIT INTEGER(A-Z)
      CØMMØN /MØV/ BUFFER(1800),BUFPTR,BUFSIZ,INPUT(90)
      IF (BUFPTR .LT. BUFSIZ) GØ TØ 100
      CALL BUFFERIN(120,1,BUFFER,BUFSIZ,ISTAT)
      BUFPTR = 0
  100 IF (LENG .EQ. 0) GØ TØ 300
```

```
      DO 200 I=1,LENG
  200 INPUT(I) = BUFFER(BUFPTR+I)
  300 BUFPTR = BUFPTR + 90
      END


      SUBROUTINE OMOVE
C     WRITES LOGICAL 90-WORD RECORD TO ARCHIVE TAPE
      IMPLICIT INTEGER(A-Z)
      COMMON /MOV/ BUFFER(1800),BUFPTR,BUFSIZ,INPUT(90)
      IF (BUFPTR .LT. BUFSIZ) GO TO 100
      CALL BUFFEROUT(120,1,BUFFER,BUFSIZ,ISTAT)
      BUFPTR = 0
  100 DO 200 I=1,90
  200 BUFFER(BUFPTR+I) = INPUT(I)
      BUFPTR = BUFPTR + 90
      END


      SUBROUTINE DOTSYS
      IMPLICIT INTEGER(A-Z)
      COMMON /BUFF/ BUFFER(10)
      EQUIVALENCE (BUFFER(1),BUFF1),(BUFFER(2),BUFF2,RBUFF(1))
      DIMENSION RBUFF(9)
      COMMON /CTAB/ CTABLE(3,750),SIGNS(750),BRANCH(750),NCT,TSIGN(4)
      COMMON /ALPH/ SYMBOL(64),CCLASS(64),SINGSG(64),EXTENT(64),
     *   COMPB(64),ISOLET(64),NALPH
      COMMON /LOGIC/ DECIS(15,25),COND(15,10),TRANS(10,25),NONTER(2),
     *   RCCLAS(4,2),NDTC,NIC,NNT
      COMMON /STV/ STVAR(10),NSV
      COMMON /SPAC/ SPACE
      COMMON /S/ LETS
      COMMON /N/ LETN
      COMMON /COMP/ COMPUT
      LOGICAL COMPUT
      COMMON /RCC/ RCC
C     FOLLOWING COMMONS NOT USED IN MAIN PROG BUT NEEDED FOR OVERLAY
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      COMMON /IN/ INPTR,PRIORS,SCOUNT
```

```
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
      COMMON /OPTS/ EMBOPT,PFOUT
      COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),PCHAR(64),MCODE(64)
      COMMON /MEMB/ EMBOUT(20),MEMBCR,MEMBPG,MIDLE
      DIMENSION BLANKS(9)
      EQUIVALENCE (BLANKS,EMBOUT(12))
      COMMON /CPR/ CPRINT
      COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
      COMMON /LC/ LCOUNT
      COMMON /VAL/ VALUE(64),BVAL(10)
C
      DATA LETA,LETF,LETG,LETI,LETN,LETO,LETR,LETS,LETT,LETY,STAR,
     *   HYPHEN,ITALIC,SPACE,LETSIG,CAPSIG,ACCENT,OPBRAK,CLBRAK/
     *   'A','F','G','I','N','O','R','S','T','Y','*','-','_',' ','=',
     *   '¬','¢','(',')'/
      DATA DIGIT /26,1,3,9,25,17,11,27,19,10/
      DATA RCC / '|' /
      DATA EMBOUT(1),EMBOUT(11) /8ZCA120000,8Z0A3C0000/
      DATA EMBOUT(1),EMBOUT(11) / '    ','    ' /
      DATA TBB/'    '/
      DATA BLANKS / 9*'    ' /
    1 FORMAT(' NEW CHAR ',Z8)
      OUTPUT 'DOTSYS III-F'
C
      CALL SEGLOD(3)
      CALL INIT
      CALL SEGLOD(4)
      CALL SHIFT(10)
      CALL PCLEAR
      CALL RESET
C
C     IDENTIFY LEFTMOST CHARACTER OF BUFFER
  200 LETTER = BVAL(1)
      IF (LETTER .NE. 0) GO TO 600
      IF (BUFF1 .NE. RCC .AND. BUFF1 .NE. STAR) GO TO 9800
      CALL OPSIGN(98)
C     RUN WILL STOP WITH ABOVE CALL
C
  600 IF (ISOLET(LETTER) .NE. 1) GO TO 3200
```

```
C
C     INSERT LETTER SIGN BEFØRE SINGLE LETTER IN BRACKETS, QUØTES, ETC.
      IF (BUFF1 .EQ. ITALIC .AND. (BUFF2 .EQ. LETA .ØR. BUFF2 .EQ. LETI
     *   .ØR. BUFF2 .EQ. LETØ .ØR. BUFF2 .EQ. ITALIC)) GØ TØ 3200
      N = 1
      K = 1
  800 T = RBUFF(N)
      IF (T .EQ. LETSIG) GØ TØ 3200
      IF (T .EQ. CAPSIG .ØR. T .EQ. ACCENT) GØ TØ 2200
      IF (T .EQ. ITALIC) GØ TØ 2400
      X = BVAL(N+1)
      IF (X .NE. 0) GØ TØ 1200
      WRITE(108,1) T
 1200 IF (CCLASS(X) .NE. 1) GØ TØ 3200
      M = N + 1
      IF (BUFF1 .EQ. ITALIC) GØ TØ 2600
      IF (SYMBØL(LETTER) .EQ. ØPBRAK) GØ TØ 1400
      IF (SYMBØL(LETTER) .EQ. RBUFF(M)) GØ TØ 1600
      GØ TØ 3200
C
 1400 IF (RBUFF(M) .NE. CLBRAK) GØ TØ 3200
 1600 DØ 1800 I=1,K
      RBUFF(M) = RBUFF(N)
      BVAL(M+1) = BVAL(N+1)
      N = N - 1
 1800 M = M - 1
      CALL SHIFT(M)
      BUFF1 = LETSIG
      BVAL(1) = ICH(VALUE,2Z7E)
      GØ TØ 200
C
 2200 K = K + 1
 2400 N = N + 1
      GØ TØ 800
C
 2600 X = BVAL(M+1)
      IF (X .EQ. 0) X = ICH(VALUE,2Z5C)
 3000 IF (CCLASS(X) .EQ. 1) GØ TØ 3200
      IF (N .GT. K) CALL SHIFT(1)
      BUFF1 = LETSIG
```

```
      BVAL(1) = ICH(VALUE,2Z7E)
      GØ TØ 200
C
C     CØNTRACTIØN TABLE SEARCH
C     INDEX PØINTS TØ CØNTRACTIØN TABLE ENTRY UNDER CØNSIDERATIØN
C
 3200 INDEX = EXTENT(LETTER)
      IF (INDEX .GT. NCT) GØ TØ 9200
      NXTLEV = 1
C
C     TEST FØR MATCH BETWEEN CURRENT TABLE ENTRY AND BUFFER
 3400 I12 = INDEX * 12 - 13
S          LW,1      NXTLEV      PØINTR
S3450      LW,7      I12
S          AW,7      1
S          LB,9      CTABLE,7
S          CB,9      RCC
S          BE        3500S
S          LW,7      1
S          MI,7      4
S          CB,9      BUFFER,7
S          BNE       3900S
S          AI,1      1
S          CI,1      9
S          BLE       3450S
S          AI,1      -1
S3500      LW,2      INDEX
S          B         5000S
C
C       REGISTERS  1=PØINTR, 2=INDEX, 3=BRI, 4=-BRI
S3800      LW,1      PØINTR
S3900      LW,2      INDEX
S          LW,3      BRANCH-1,2
S          LCW,4     3
S          BLZ       4600S
S          CW,1      NXTLEV
S          BNE       4400S
S4000      CW,4      NXTLEV
S          BL        4400S
S          AI,2      1
S4200      LW,3      BRANCH-1,2
```

```
S          LCW,4     3
S          BGFZ      4000S
S          LW,2      3
S          B         4200S
S4400      STW,4     NXTLEV
S          CI,4      1
S          BLE       9200S
S          AI,2      1
S          STW,2     INDEX
S          B         3400S
S4600      CW,1      NXTLEV
S          BE        4800S
S          LW,7      2
S          AI,7      1
S          CW,3      7
S          BE        $+2
S          MTW,1     NXTLEV
S          AI,2      1
S          STW,2     INDEX
S          B         3400S
S4800      STW,3     2
S          STW,2     INDEX
S          B         3400S
S5000      LW,7      2
S          MI,7      12
S          AI,7      -1
S          LB,9      CTABLE,7
S          CB,9      SPACE
S          BE        6100S
S          STB,9     TBB
S          LW,9      BVAL,1
S          STW,9     N
S          BEZ       3800S
S          STW,1     PØINTR
S          STW,2     INDEX
C
 5400 DØ 6000 K=1,NNT
      IF (TBB .NE. NØNTER(K)) GØ TØ 6000
      DØ 5800 J=1,4
      IF (CCLASS(N) .EQ. RCCLAS(J,K)) GØ TØ 6200
```

```
 5800 IF (RCCLAS(J,K) .EQ. 99) GO TO 3800
      GO TO 3800
 6000 CONTINUE
      GO TO 3800
C
C     RIGHT CONTEXT IS O.K. - NOW CHECK INPUT CLASS FOR COMPATIBILITY
C     WITH CURRENT VALUE OF STATE VARIABLES
S6100    STW,1       POINTR
S        STW,2       INDEX
 6200 CONTINUE
S        LW,7        INDEX
S        MI,7        12
S        AI,7        -2
S        LB,9        CTABLE,7
S        STW,9       TCLASS
      DO 7000 K=1,NDTC
      DKT = DECIS(K,TCLASS)
      IF (DKT .EQ. HYPHEN) GO TO 7000
      IF (DKT .EQ. LETG) GO TO 7600
      IF (DKT .EQ. LETF) GO TO 3800
      DO 6800 N=1,NSV
      CKN = COND(K,N)
 6800 IF (CKN .NE. HYPHEN .AND. CKN .NE. STVAR(N)) GO TO 7400
      GO TO 7200
 7000 CONTINUE
C
 7200 IF (DKT .EQ. LETY) GO TO 7600
      GO TO 3800
 7400 IF (DKT .EQ. LETY) GO TO 3800
C
C     CONTRACTION TABLE ENTRY IS ACCEPTED
C     SEND APPROPRIATE SIGNS TO STACKER AND SHIFT BUFFER LEFT BY
C     NUMBER OF CHARACTERS INDICATED IN CONTRACTION TABLE ENTRY
 7600 CONTINUE
S        LW,7        INDEX
S        MI,7        12
S        AI,7        -3
S        LB,9        CTABLE,7
S        STW,9       SHIFNO
      CALL SHIFT(SHIFNO)
```

```
      I = SIGNS(INDEX)
      DØ 8200 J=0,3
S        LW,7       J
S        LB,9       I,7
S        CI,9       99
S        BE         8400S
S        STW,9      ØUTSIG
 8200 CALL ØPSIGN(ØUTSIG)
C
C     ADJUST STATE VARIABLES ACCØRDING TØ TRANSITIØN TABLES
 8400 I = TCLASS
      DØ 9000 N=1,NSV
      TNI = TRANS(N,I)
      IF (TNI .EQ. HYPHEN) GØ TØ 9000
      IF (TNI .NE. LETS .AND. (TNI .NE. LETT .ØR. STVAR(N) .NE. LETN))
     *   GØ TØ 8800
      STVAR(N) = LETY
      GØ TØ 9000
 8800 STVAR(N) = LETN
 9000 CØNTINUE
      GØ TØ 200
C
C     NØ CØNTRACTIØN TABLE ENTRY IS ACCEPTABLE - ØUTPUT SINGLE CHARACTER
C     ØBTAINED FRØM ALPHABET TABLE
 9200 IF (.NØT. CØMPUT) GØ TØ 9400
      ØUTSIG = CØMPB(LETTER)
      GØ TØ 9600
 9400 ØUTSIG = SINGSG(LETTER)
 9600 CALL SHIFT(1)
      CALL ØPSIGN(ØUTSIG)
      TCLASS = CCLASS(LETTER)
      GØ TØ 8400
C
C     LEFTMØST CHARACTER ØF BUFFER DØES NØT ØCCUR IN ALPHABET TABLE
C     ØUTPUT ERRØR MESSAGE
 9800 WRITE(108,1) BUFF1
      BUFF1 = STAR
      ØVAL(1) = ICH(VALUE,2Z5C)
      GØ TØ 200
      END
```

```
      INTEGER FUNCTION CHAR(ARRAY,I,J)
      IMPLICIT INTEGER(A-Z)
      DATA ADDEND /6Z404040/
      CHAR = ISL(ICH(ARRAY,12*(I-1)+J),24) + ADDEND
      END


      SUBROUTINE INIT
C
C     INITIALISATION SUBROUTINE
      IMPLICIT INTEGER(A-Z)
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      EQUIVALENCE(HEADS1,HEADS(1)),(TAILS1,TAILS(1)),(CURRS1,CURRS(1))
      EQUIVALENCE(HEADS2,HEADS(2)),(TAILS2,TAILS(2)),(CURRS2,CURRS(2))
      COMMON /IN/ INPTR,PRIORS,SCOUNT
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
      COMMON /OPTS/ EMBOPT,PFOUT
      COMMON /SIGNS/DOTS14(64),DOTS25(64),DOTS36(64),PCHAR(64),MCODE(64)
      COMMON /MEMB/ EMBOUT(20),MEMBCR,MEMBPG,MIDLE
      COMMON /CPR/ CPRINT
      COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
      COMMON /SPAC/ SPACE
      COMMON /CHEK/ CARDNO,PREVNO,ECHO,SEQERR
      COMMON /CTAB/ CTABLE(3,750),SIGNS(750),BRANCH(750),NCT,TSIGN(4)
      COMMON /ALPH/ SYMBOL(64),CCLASS(64),SINGSG(64),EXTENT(64),
     *   COMPB(64),ISOLET(64),NALPH
      COMMON /LC/ LCOUNT
      COMMON /LOGIC/ DECIS(15,25),COND(15,10),TRANS(10,25),NONTER(2),
     *   RCCLAS(4,2),NDTC,NIC,NNT
      COMMON /STV/ STVAR(10),NSV
      COMMON /RCC/ RCC
      COMMON /VAL/ VALUE(64),BVAL(10)
      LOGICAL EMBOPT,PFOUT,SEQERR,ECHO,PAGINA,CPRINT
      DIMENSION LARRAY(10)
      DATA LETE,LETL,LETM,LETP / 'E','L','M','P' /
      DATA CTMAX /1000/
C
    1 FORMAT(80X)
```

```
    2 FØRMAT(' ** TABLE SEQUENCE ERRØR')
    3 FØRMAT(A1,71X,I8)
    4 FØRMAT(A1,16X,3(I2,X),3X,I2,41X,I8)
    5 FØRMAT(A1,3A4,4X,2(I2,X),4I2,41X,I8)
    6 FØRMAT(17X,I2,53X,I8)
    7 FØRMAT(12X,A1,10X,4I2,41X,I8)
    8 FØRMAT(25A1,47X,I8)
    9 FØRMAT(NA1,NA1,I)
   10 FØRMAT(3A2,A3,A1,62X,I8)
   11 FØRMAT(A1,14X,5A1,52X,I8)
   12 FØRMAT(A1,30X,4I1,37X,I8)
   13 FØRMAT(' TØØ MANY CØNTRACTIØN TABLE ENTRIES')
C
      WRITE(108,1)
      INPTR = 81
      SEQERR = .FALSE.
      PREVNØ = 0
      ECHØ = .TRUE.
      READ(131,3) T,CARDNØ
      ECHØ = T .EQ. LETE
      CALL SEQCHK
      DØ 100 N=1,64
  100 VALUE(N) = 0
C
C     SET UP STACKS
      DØ 200 N=1,19
      NØELTS(N) = 0
      SPCØNT(N) = 0
      PVSTAK(N) = N - 1
  200 NXSTAK(N) = N + 1
      NXSTAK(19) = 0
      NXSTAK(1) = 0
      PVSTAK(2) = 0
      HFSTAK = 3
      CURTYP = 1
      HEADS1 = 1
      TAILS1 = 1
      CURRS1 = 1
      HEADS2 = 2
      TAILS2 = 2
```

```
      CLRPS2 = 2
      HS = 1
      TS = 1
      CS = 1
      PVSTAK(3) = 0
      NXSTAK(2) = 0
C
      DO 600 N = 1,9
      TABTYP(N) = LETL
  6C0 TABCLM(N) = 2 * N + 1
      LCOUNT = 0
      OUTPTR = 1
      TABULA = 0
      SCOUNT = 0
      TEMPL = 0
      STKIND = 2
      CPRINT = .FALSE.
      PRIORS = 0
C
C     READ ALPHABET TABLE
      DO 800 N=1,65
      EXTENT(N) = 999
      READ(131,4) SYMBOL(N),CCLASS(N),COMPB(N),SINGSG(N),ISOLET(N),
     *   CARDNO
      CALL SEQCHK
      IF (CCLASS(N) .EQ. 99) GO TO 1000
      T = SYMBOL(N)
C     ICH(VALUE,ICH(SYMBOL(N),1)) = N
S        LB,7       T
S        AI,7       -1
S        LW,9       N
S        STB,9      VALUE,7
  8C0 CONTINUE
 10C0 NALPH = N
      I = 1
C
C     FILL CONTRACTION TABLE
      DO 1800 N=1,CTMAX
      READ(131,5) LET1,(CTABLE(J,N),J=1,3),ICLASS,SHIFNO,
     *   (TSIGN(J),J=1,4),CARDNO
```

```
C     PUT ICLASS IN BYTE 11 OF CTABLE ENTRY
S        LW,7       N
S        MI,7       12
S        AI,7       -2
S        LW,9       ICLASS
S        STB,9      CTABLE,7
S        AI,7       -1
S        LW,9       SHIFNO
S        STB,9      CTABLE,7
      CALL SEQCHK
C     PACK SIGNS INTO SIGNS(N)
      DO 1100 J=0,3
S        LW,7       J
S        LW,9       TSIGN,7
S1100    STB,9      T,7
      SIGNS(N) = T
      IF (LET1 .EQ. TEMPL) GO TO 1400
      TEMPL = LET1
      EXTENT(I) = N
      IF (SYMBOL(I) .EQ. TEMPL .OR. ICLASS .EQ. 99) GO TO 1200
      ECHO = .TRUE.
      SEQERR = .TRUE.
      WRITE(108,2)
 1200 I = I + 1
 1400 CONTINUE
 1800 IF (ICLASS .EQ. 99) GO TO 2000
      WRITE(108,13)
      STOP 'ERROR'
C
 2000 NCT = N - 1
      J = I - 1
      EXTENT(I) = EXTENT(J) + 1
      MAXEXT = J
C
C     CONTRACTION TABLE SEARCH SET-UP ALGORITHM
      T = NCT + 1
      DO 2200 I=1,T
 2200 BRANCH(I) = 0
      DO 4000 I=1,MAXEXT
      J = 1
```

```
      LOWNO = EXTENT(I)
      J = I + 1
      HIGHNO = EXTENT(J) - 1
      VHIGH = HIGHNO
 2600 II = LOWNO + 1
      IF (II .GT. HIGHNO) GO TO 3600
      IF (CHAR(CTABLE,LOWNO,N) .EQ. RCC) GO TO 3000
 2800 IF (II .GT. HIGHNO) GO TO 3600
      IF (CHAR(CTABLE,LOWNO,N) .NE. CHAR(CTABLE,II,N)) GO TO 3000
      II = II + 1
      GO TO 2800
 3000 LARRAY(N) = II
      BRANCH(LOWNO) = II
      BRANCH(II) = N
      N = N + 1
      LOWNO = LOWNO + 1
      HIGHNO = II - 1
 3200 IF (LOWNO .LE. HIGHNO) GO TO 2600
      N = N - 1
 3400 IF (N .EQ. 0) GO TO 4000
      IF (N .LE. 1) GO TO 3500
      HIGHNO = LARRAY(N-1) - 1
      GO TO 3200
 3500 HIGHNO = VHIGH
      GO TO 3200
 3600 IF (II .NE. LOWNO + 1) GO TO 3800
      BRANCH(LOWNO) = - BRANCH(II)
      LOWNO = LOWNO + 1
      BRANCH(II) = N
      GO TO 3400
 3800 LARRAY(N) = II
      N = N + 1
      BRANCH(LOWNO) = -N
      LOWNO = LOWNO + 1
      GO TO 2600
 4000 CONTINUE
C
C     READ RIGHT-CONTEXT TABLE
      READ(131,6) NNT,CARDNO
      CALL SEQCHK
```

```
      DØ 4200 N=1,NNT
      READ(131,7) MØNTER(N),(RCCLAS(I,N),I=1,4),CARDNØ
 4200 CALL SEQCHK
C
C     READ STATE TABLES
      READ(131,6) NSV,CARDNØ
      CALL SEQCHK
      READ(131,6) NIC,CARDNØ
      CALL SEQCHK
      DØ 4400 I=1,NSV
      READ(131,8) (TRANS(I,J),J=1,25),CARDNØ
 4400 CALL SEQCHK
      READ(131,6) NDTC,CARDNØ
      CALL SEQCHK
      DØ 4600 N =1,NDTC
      READ(131,9) NIC,(DECIS(N,I),I=1,NIC),NSV,
     *   (CØND(N,I),I=1,NSV),CARDNØ
 4600 CALL SEQCHK
C
C     READ SIGN TABLE
      DØ 5000 N = 1,64
      READ(131,10) DØTS14(N),DØTS25(N),DØTS36(N),PCHAR(N),MCØDE(N),
     *   CARDNØ
 5000 CALL SEQCHK
C
C     READ CØNTRØL CARDS
      READ(131,3) T,CARDNØ
      CALL SEQCHK
      PFØUT = T .EQ. LETP
      READ(131,11) T,MEMBØN,MEMBØF,MIDLE,MEMBCR,MEMBPG,CARDNØ
      CALL SEQCHK
      EMBØPT = T .EQ. LETM
      READ(131,6) MUTL,CARDNØ
      CALL SEQCHK
      READ(131,6) LPG,CARDNØ
      CALL SEQCHK
      READ(131,12) T,(CDIGIT(I),I=1,4),CARDNØ
      CALL SEQCHK
      PAGINA = T .EQ. LETP
      IF (SEQERR) STØP 'SEQUENCE ERRØR'
```

```
      END

      SUBRØUTINE SEQCHK
C     SEQUENCE CHECK FØR TABLES INPUT
      IMPLICIT INTEGER(A-Z)
      CØMMØN /CHEK/ CARDNØ,PREVNØ,ECHØ,SEQERR
      LØGICAL SEQERR,ECHØ
    1 FØRMAT(' **FØLLOWING CARDS ØUT ØF SEQUENCE ',I8)
C
      IF (CARDNØ .LE. PREVNØ) GØ TØ 100
      PREVNØ = CARDNØ
      IF (.NØT. ECHØ) RETURN
      GØ TØ 200
  100 SEQERR = .TRUE.
      PREVNØ = 0
  200 WRITE(108,1) CARDNØ
      END


      SUBRØUTINE TABDIS
      IMPLICIT INTEGER(A-Z)
      CØMMØN /STV/ STVAR(10),NSV
      CØMMØN /LØGIC/ DECIS(15,25),CØND(15,10),TRANS(10,25),NØNTER(2),
     *   RCCLAS(4,2),NDTC,NIC,NNT
   10 FØRMAT(///16X,'DECISIØN TABLE'//'-CØLUMN')
   11 FØRMAT(///16X,'RIGHT CØNTEXT TABLE'//' NØN-TERMINAL    INPUT CLASSE
     *S'//)
   12 FØRMAT(6X,A1,8X,4(I2,2X))
   13 FØRMAT(19X,20(2X,I2))
   14 FØRMAT(/'-INPUT CLASS')
   15 FØRMAT(17X,I2,3X,20(A1,3X))
   16 FØRMAT(/'-STATE VARIABLE')
   17 FØRMAT(///16X,'TRANSITIØN TABLE'//'-STATE VARIABLE')
   18 FØRMAT(//)
      WRITE(108,11)
      DØ 6200 K=1,NNT
 6200 WRITE(108,12) NØNTER(K),(RCCLAS(I,K),I=1,4)
      WRITE(108,10)
      WRITE(108,13) (K,K=1,NDTC)
```

```
      WRITE(108,14)
      DØ 6400 N=1,NIC
 6400 WRITE(108,15) N,(DECIS(K,N),K=1,NDTC)
      WRITE(108,16)
      DØ 6600 N=1,NSV
 6600 WRITE(108,15) N,(CØND(K,N),K=1,NDTC)
      WRITE(108,17)
      WRITE(108,13) (K,K=1,NSV)
      WRITE(108,14)
      DØ 6800 N=1,NIC
 6800 WRITE(108,15) N,(TRANS(K,N),K=1,NSV)
      WRITE(108,18)
      END


      SUBRØUTINE SHIFT(NCHARS)
C     SHIFT BUFFER LEFT BY NCHARS
      IMPLICIT INTEGER(A-Z)
      CØMMØN /IN/ INPTR,PRIØRS,SCØUNT
      CØMMØN /ØUT/ ØUTPTR,ØUTL,LINES,LPG
      CØMMØN /BUFF/ BUFFER(10)
      CØMMØN /ØPTS/ EMBØPT,PFØUT
      LØGICAL EMBØPT,PFØUT
      CØMMØN /SPAC/ SPACE
      CØMMØN /CPR/ CPRINT
      LØGICAL CPRINT
      CØMMØN /RCC/ RCC
      DIMENSIØN TEXT(20)
      CØMMØN /VAL/ VALUE(64),BVAL(10)
      DATA NXTCHR /'    '/
    2 FØRMAT(X,20A4)
    3 FØRMAT(' INPUT RECØRD LESS THAN 20 WØRDS')
C
      IF (NCHARS .EQ. 0) RETURN
      DØ 1400 I=1,NCHARS
S        LI,7        -9
S50      LW,9        BUFFER+10,7
S        STW,9       BUFFER+9,7
S        LW,9        BVAL+10,7
S        STW,9       BVAL+9,7
```

```
S          BIR,7      50S
  100 IF (INPTR .LE. 80) GØ TØ 200
      IF (SCØUNT .GT. 0) GØ TØ 800
      CALL BUFFERIN(131,1,TEXT,20,ISTAT,N)
      IF (ISTAT .EQ. 3) GØ TØ 800
      IF (N .NE. 20) WRITE(108,3)
      INPTR = 1
      IF (.NØT. PFØUT) GØ TØ 200
      WRITE(108,2) TEXT
      CPRINT = .TRUE.
  200 CØNTINUE
S          LW,7       INPTR
S          AI,7       -1
S          LB,9       TEXT,7
S          STB,9      NXTCHR
S          MTW,1      INPTR
S          CB,9       SPACE
S          BNE        400S
      IF (PRIØRS .EQ. 0) GØ TØ 100
      PRIØRS = PRIØRS - 1
      GØ TØ 600
  400 PRIØRS = 1
  600 IF (NXTCHR .EQ. RCC) GØ TØ 100
      GØ TØ 1200
  800 SCØUNT = SCØUNT + 1
      IF (SCØUNT .EQ. 1 .AND. PRIØRS .GT. 0) GØ TØ 1000
      NXTCHR = RCC
      GØ TØ 1200
 1000 NXTCHR = SPACE
 1200 BUFFER(10) = NXTCHR
S          LB,7       NXTCHR
S          AI,7       -1
S          LB,9       VALUE,7
S          STW,9      BVAL+9
 1400 CØNTINUE
      END
```

```
      SUBROUTINE OPSIGN(OUTSIG)
C     STORE SIGN ON CURRENT STACK
      IMPLICIT INTEGER(A-Z)
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      EQUIVALENCE(HEADS1,HEADS(1)),(TAILS1,TAILS(1)),(CURRS1,CURRS(1))
      EQUIVALENCE(HEADS2,HEADS(2)),(TAILS2,TAILS(2)),(CURRS2,CURRS(2))
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /PAG/ CDIGIT(4),DIGIT(10),PAGINA
      LOGICAL PAGINA
      COMMON /BUFF/ BUFFER(10)
      COMMON /OPTS/ EMBOPT,PFOUT
      LOGICAL EMBOPT,PFOUT
      COMMON /S/ LETS
      COMMON /TTL/ TTLENG
      COMMON /COMP/ COMPUT
      COMMON /INDENT/ MARGIN,INHEAD
      LOGICAL INHEAD,COMPUT
      COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
      COMMON /DUB/ DOUBLE
      LOGICAL DOUBLE
      DATA CCHIGH,SCHIGH/80,89/
C
C     CHARACTER TEST
      IF (OUTSIG - 64) 800,400,200
  200 IF (OUTSIG .LE. CCHIGH) GO TO 1400
      IF (OUTSIG .LE. SCHIGH) GO TO 1800
      GO TO 2000
C
C     OUTPUT SPACE
  400 IF (STKIND .NE. 2) GO TO 600
      CALL CLEAR
      RETURN
  600 CALL STAKTB
      RETURN
C
C     OUTPUT ORDINARY CHARACTER
  800 IF (NOELTS(CS) .NE. 30) GO TO 1000
      CALL STAKTB
      NOELTS(CS) = 1
```

```
      GO TO 1200
 1000 NOELTS(CS) = NOELTS(CS) + 1
 1200 ELT(CS,NOELTS(CS)) = OUTSIG
      RETURN
C
C     CARRIAGE CONTROL
 1400 IF (NOELTS(CS) .NE. 0) CALL STAKTB
      NOELTS(CS) = 30
      SPCENT(CS) = OUTSIG
      IF (OUTSIG .NE. 72) GO TO 1600
      CALL DECODE(1,ELT(CS,3))
      CALL SHIFT(1)
 1500 IF (STKIND .EQ. 2) STKIND = 5
      RETURN
 1600 IF (OUTSIG .NE. 68 .AND. OUTSIG .NE. 71) RETURN
 1700 CALL DECODE(2,ELT(CS,3))
 1750 CALL SHIFT(2)
      RETURN
C
C     STACK CONTROL
 1800 T = OUTSIG - 80
      GO TO (2400,2600,5000,1500,1900,4600,1900,3600,4000),T
 1900 RETURN
C
C     MODE CONTROL
 2000 IF (OUTSIG .EQ. 93) GO TO 5800
      IF (OUTSIG .EQ. 95) GO TO 4800
      IF (OUTSIG .EQ. 98) GO TO 5600
      IF (OUTSIG .EQ. 97) COMPUT = .NOT. COMPUT
      IF (OUTSIG .EQ. 90) DOUBLE = .NOT. DOUBLE
      RETURN
C
C     BEGIN HEADING
 2400 IF (NOELTS(CS) .NE. 0) CALL CLEAR
      CALL CARRIJ(65)
      STKIND = 4
      RETURN
C
C     END HEADING
 2600 H = HEADS1
```

```
      M = 0
 2800 M = M + NØELTS(N)
      IF (SPCØNT(N) .EQ. 0 .AND. NØELTS(N) .GT. 0) M = M + 1
      IF (N .EQ. TAILS1) GØ TØ 3000
      N = NXSTAK(N)
      GØ TØ 2800
C
 3000 ØUTPTR = (ØUTL - M) / 2 + 2
      IF (ØUTPTR .LT. 1) ØUTPTR = 1
      STKIND = 2
      INHEAD = .TRUE.
      CALL CLEAR
      INHEAD = .FALSE.
      CALL CARRIJ(65)
      RETURN
C
C     BEGIN TITLE
 3600 CURTYP = 2
      HS = HEADS2
      TS = TAILS2
      CS = CURRS2
      STKIND = 4
      M = HS
      IF (NØELTS(M) .EQ. 0) RETURN
C     CLEAR ØUT PREVIØUS TITLE
      NØELTS(M) = 0
 3800 IF (HS .EQ. TS) RETURN
      CALL FRSTAK(PVSTAK,NXSTAK,TS,TAILS)
      CURRS2 = HEADS2
      GØ TØ 3800
C
C     END TITLE
 4000 TTLENG = 0
      N = HEADS2
 4200 TTLENG = TTLENG + NØELTS(N)
      IF (SPCØNT(N) .EQ. 0 .AND. NØELTS(N) .GT. 0) TTLENG = TTLENG + 1
      IF (N .EQ. TAILS2) GØ TØ 4400
      N = NXSTAK(N)
      GØ TØ 4200
C
```

```
 4400 CURTYP = 1
      HS = HEADS1
      TS = TAILS1
      CS = CURRS1
      STKIND = 2
      RETURN
C
C     UNIT OF MEASURE
 4600 SPCENT(CS) = 1
C
C     INTERCHANGE LAST TWO ROWS OF CURRENT STACK
      N = PVSTAK(TS)
      IF (N .EQ. 0) GO TO 4700
      M = PVSTAK(N)
      PVSTAK(TS) = M
      NXSTAK(TS) = N
      PVSTAK(N) = TS
      NXSTAK(N) = 0
      TS = N
      TAILS(CURTYP) = N
      CS = N
      CURRS(CURTYP) = N
      IF (N .NE. HS) GO TO 4650
      HS = PVSTAK(N)
      HEADS(CURTYP) = HS
      GO TO 4700
 4650 NXSTAK(M) = PVSTAK(N)
C
 4700 IF (BUFFER(1) .EQ. LETS) CALL SHIFT(1)
      RETURN
C
C     MARGIN RESET
 4800 CALL DECODE(2,MARGIN)
      GO TO 1750
C
C     RESTART
 5000 DO 5200 N=1,3
 5200 CDIGIT(N) = 0
      CDIGIT(4) = 1
      CALL RESET
```

```
      RETURN
 5500 CALL CARRIJ(65)
      ENDFILE 121
      STOP 'OK'
 5800 CALL DECODE(1,T)
      TABTYP(T) = BUFFER(2)
      CALL SHIFT(2)
      CALL DECODE(2,TABCLM(T))
      CALL SHIFT(2)
      RETURN
      END


      SUBROUTINE CARRIJ(OUTSIG)
      IMPLICIT INTEGER(A-Z)
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /LC/ LCOUNT
      COMMON /DUB/ DOUBLE
      LOGICAL DOUBLE
C
      IF (LCOUNT .GE. LPG) CALL PAGEHD
      IF (OUTPTR .EQ. 1 .AND. (OUTSIG .EQ. 65 .OR. OUTSIG .EQ. 67))
     *   GO TO 600
      CALL OUTOUT(OUTSIG)
      IF (DOUBLE .AND. LCOUNT .LT. LPG) CALL OUTOUT(OUTSIG)
  600 IF (OUTSIG .NE. 67) GO TO 800
      OUTPTR = 3
      RETURN
  800 OUTPTR = 1
      END


      SUBROUTINE CLEAR
C     CLEAR CURRENT STACK
      IMPLICIT INTEGER(A-Z)
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
      COMMON /OUT/ OUTPTR,OUTL,LINES,LPG
      COMMON /TAB/ TABTYP(9),TABCLM(9),TABULA,LETR,LETD
      COMMON /INDENT/ MARGIN,INHEAD
```

```
      LØGICAL INHEAD
C
  100 PS = HS
      NPS = NØELTS(PS)
  400 IF (SPCØNT(PS) .LT. 64) GØ TØ 3000
      ØUTSIG = SPCØNT(PS)
      EP3 = ELT(PS,3)
      IF (ØUTSIG .NE. 68) GØ TØ 600
C     ØNE-TIME TABULATIØN
      TABULA = EP3
      GØ TØ 4100
  600 IF (ØUTSIG .NE. 71) GØ TØ 1800
C     SKIP LINES
      CALL CARRIJ(65)
      CALL PCLEAR
      DØ 1600 M=1,EP3
      ØUTPTR = 0
 1600 CALL CARRIJ(71)
      GØ TØ 4200
C
C     PRESET TABULATIØN
 1800 IF (ØUTSIG .NE. 72) GØ TØ 2800
      M = ELT(PS,3)
      XX = NXSTAK(PS)
      IF (TABTYP(M) .EQ. R) GØ TØ 2000
      IF (TABTYP(M) .EQ. D) GØ TØ 2200
      TABULA = TABCLM(M)
      GØ TØ 4100
C
 2000 TABULA = TABCLM(M) - NØELTS(XX) + 1
      GØ TØ 4100
C
 2200 T = NØELTS(XX)
      DØ 2400 XYZ=1,T
 2400 IF (ELT(XX,XYZ) .EQ. 40) GØ TØ 2600
 2600 TABULA = TABCLM(M) - XYZ + 1
      GØ TØ 4100
C
 2800 CALL CARRIJ(ØUTSIG)
      GØ TØ 4200
```

```
C
 3000 IF (NPS .EQ. 0) GØ TØ 4200
      IF (ØUTPTR + NPS .LT. ØUTL + 2) GØ TØ 3800
      ØUTPTR = ØUTL + 1
      CALL CARRIJ(0)
      IF (MARGIN .LE. 0) GØ TØ 3400
      DØ 3200 N=1,MARGIN
 3200 CALL MØVEEL(64)
 3400 IF (.NØT. INHEAD) GØ TØ 3800
      DØ 3600 N=1,3
 3600 CALL MØVEEL(64)
 3800 DØ 4000 M=1,NPS
 4000 CALL MØVEEL(ELT(PS,M))
      IF (SPCØNT(PS) .NE. 1) CALL MØVEEL(64)
      GØ TØ 4200
C
C     TABULATIØN
 4100 TABULA = TABULA - ØUTPTR
      IF (TABULA .LE. 0) GØ TØ 4120
      DØ 4110 N=1,TABULA
 4110 CALL MØVEEL(64)
      GØ TØ 4140
 4120 TABULA = TABULA + ØUTPTR - 1
      CALL CARRIJ(65)
      DØ 4130 N=1,TABULA
 4130 CALL MØVEEL(64)
 4140 TABULA = 0
C
 4200 IF (HS .EQ. TS) GØ TØ 4800
      CALL FRSTAK(NXSTAK,PVSTAK,HS,HEADS)
      GØ TØ 100
 4800 NØELTS(HS) = 0
      SPCØNT(HS) = 0
      CURRS(CURTYP) = HS
      CS = HS
      END
```

```
      SUBRØUTINE FRSTAK(STAK,STAK1,HT,HTS)
      IMPLICIT INTEGER(A-Z)
      CØMMØN /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCØNT(19),NØELTS(19),PVSTAK(19),NXSTAK(19)
      DIMENSIØN STAK(19),STAK1(19),HTS(2)
C
      NØELTS(HT) = 0
      SPCØNT(HT) = 0
      IF (STAK(HT) .EQ. 0) RETURN
      M = HT
      N = HFSTAK
      HT = STAK(M)
      HTS(CURTYP) = HT
      HFSTAK = M
      STAK1(HT) = 0
      NXSTAK(M) = N
      END


      SUBRØUTINE MØVEEL(X)
C     MØVE SIGN TØ ØUTPUT BUFFER
      IMPLICIT INTEGER(A-Z)
      CØMMØN /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCØNT(19),NØELTS(19),PVSTAK(19),NXSTAK(19)
      CØMMØN /ØUT/ ØUTPTR,ØUTL,LINES,LPG
      CØMMØN /SIGNS/DØTS14(64),DØTS25(64),DØTS36(64),PCHAR(64),MCØDE(64)
      CØMMØN /ØWAREA/ BRAILL(3,40),PRØØF(40),CØDE(40)
      CØMMØN /ØPTS/ EMBØPT,PFØUT
      LØGICAL EMBØPT,PFØUT
C
      CØDE(ØUTPTR) = X
      IF (.NØT. PFØUT) GØ TØ 200
      IF (X .EQ. 0) X = 64
      BRAILL(1,ØUTPTR) = DØTS14(X)
      BRAILL(2,ØUTPTR) = DØTS25(X)
      BRAILL(3,ØUTPTR) = DØTS36(X)
      PRØØF(ØUTPTR) = PCHAR(X)
  200 ØUTPTR = ØUTPTR + 1
      END
```

```
      SUBRØUTINE ØUTØUT(ØUTSIG)
C     WRITE ØUTPUT BUFFER
      IMPLICIT INTEGER(A-Z)
      CØMMØN /ØUT/ ØUTPTR,ØUTL,LINES,LPG
      CØMMØN /ØPTS/ EMBØPT,PFØUT
      LØGICAL EMBØPT,PFØUT
      CØMMØN /SIGNS/DØTS14(64),DØTS25(64),DØTS36(64),PCHAR(64),MCØDE(64)
      CØMMØN /BWAREA/ BRAILL(3,40),PRØØF(40),CØDE(40)
      CØMMØN /MEMB/ EMBØUT(20),MEMBCR,MEMBPG,MIDLE
      CØMMØN /CPR/ CPRINT
      LØGICAL CPRINT
      CØMMØN /LC/ LCØUNT
    1 FØRMAT(80X)
    2 FØRMAT(X,40A3)
    3 FØRMAT(40(X,I2))
    4 FØRMAT(20A4)
    5 FØRMAT(' TØØ MUCH INPUT - TRANSLATIØN INCØMPLETE')
C
      IF (.NØT. PFØUT) GØ TØ 400
      IF (.NØT. CPRINT) WRITE(108,1)
      CPRINT = .FALSE.
      DØ 200 N=1,3
  200 WRITE(108,2) (BRAILL(N,J),J=1,ØUTL)
      WRITE(108,2) PRØØF
      WRITE(108,3) (CØDE(J),J=1,ØUTL)
  400 IF (.NØT. EMBØPT) GØ TØ 1400
      DØ 600 N=1,39
      IF (N .GT. ØUTL) GØ TØ 1300
      T = CØDE(N)
      IF (T .EQ. 0) T = 64
C     ICH(EMBØUT,N+2) = ISL(MCØDE(T),-24)
S        LW,7       T
S        AI,7       -1
S        MI,7       4
S        LB,9       MCØDE,7
S        LW,7       N
S        AI,7       1
S        STB,9      EMBØUT,7
  600 CØNTINUE
 1300 CALL BUFFERØUT(132,1,EMBØUT,20,ISTAT)
```

```
      IF (ISTAT .EQ. 2) GØ TØ 1400
      WRITE(108,5)
S         CAL1,9    3              ABØRT
 1400 CALL PCLEAR
      IF (ØUTSIG .NE. 69) GØ TØ 1800
      LCØUNT = LPG
      RETURN
 1800 LCØUNT = LCØUNT + 1
      END


      SUBRØUTINE PAGEHD
      IMPLICIT INTEGER(A-Z)
      CØMMØN /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCØNT(19),NØELTS(19),PVSTAK(19),NXSTAK(19)
      CØMMØN /ØUT/ ØUTPTR,ØUTL,LINES,LPG
      CØMMØN /ØWAREA/ BRAILL(3,40),PRØØF(40),CØDE(40)
      CØMMØN /TTL/ TTLENG
      CØMMØN /LC/ LCØUNT
      CØMMØN /PAG/ CDIGIT(4),DIGIT(10),PAGINA
      LØGICAL PAGINA
      CØMMØN /ØPTS/ EMBØPT,PFØUT
      LØGICAL PFØUT,EMBØPT
      CØMMØN /MEMB/ EMBØUT(20),MEMBCR,MEMBPG,MIDLE
      CØMMØN /CPR/ CPRINT
      LØGICAL CPRINT
      CØMMØN /DUB/ DØUBLE
      LØGICAL DØUBLE
      DIMENSIØN BTEMP(3,40),CTEMP(40),PTEMP(40)
    1 FØRMAT(80X)
    2 FØRMAT(11A4)
C
      IF (.NØT. PAGINA) RETURN
      IF (PFØUT) WRITE(108,1)
      IF (.NØT. EMBØPT) GØ TØ 100
C     ICH(EMBØUT,3) = ISL(MEMBPG,-24)
S         LB,9       MEMBPG
S         LI,7       2
S         STB,9      EMBØUT,7
      DØ 50 N=4,40
```

```
C     ICH(EMBOUT,N) = ISL(MIDLE,-24)
S        LB,9      MIDLE
S        LW,7      N
S        AI,7      -1
S        STB,9     EMBOUT,7
   50 CONTINUE
      CALL BUFFEROUT(132,1,EMBOUT,20,ISTAT)
  100 LCOUNT = 0
      CPRINT = .FALSE.
      N = (OUTL - TTLENG + 2) / 2 + 1
      IF (N .LT. 1) N = 1
C     SAVE CONTENTS OF OUTPUT BUFFER
      DO 200 I=1,OUTL
      PTEMP(I) = PROOF(I)
      CTEMP(I) = CODE(I)
      DO 200 J=1,3
  200 BTEMP(J,I) = BRAILL(J,I)
      CALL PCLEAR
      OSAVE = OUTPTR
      OUTPTR = N
      N = HEADS(2)
  400 IF (NOELTS(N) .EQ. 0 .OR. SPCONT(N) .GE. 64) GO TO 800
      T = NOELTS(N)
      DO 600 XYZ = 1,T
  600 CALL MOVEEL(ELT(N,XYZ))
      IF (SPCONT(N) .EQ. 0) CALL MOVEEL(64)
  800 IF (N .EQ. TAILS(2)) GO TO 1000
      N = NXSTAK(N)
      IF (OUTPTR + NOELTS(N) .LE. OUTL - 5) GO TO 400
 1000 OUTPTR = OUTL - 4
      DO 1200 N=1,4
      IF (CDIGIT(N) .NE. 0) GO TO 1400
 1200 OUTPTR = OUTPTR + 1
 1400 CALL MOVEEL(60)
      DO 1600 N=N,4
      XYZ = CDIGIT(N) + 1
 1600 CALL MOVEEL(DIGIT(XYZ))
C
C     INCREMENT PAGE NUMBER
      DO 1800 N=4,1,-1
```

```
      CDIGIT(N) = CDIGIT(N) + 1
      IF (CDIGIT(N) .LT. 10) GO TO 2000
 1800 CDIGIT(N) = 0
 2000 CALL OUTPUT(0)
      IF (DOUBLE) CALL OUTPUT(0)
C     RESTORE OUTPUT BUFFER
      OUTPTR = OSAVE
      DO 2200 I=1,OUTL
      CODE(I) = CTEMP(I)
      PROOF(I) = PTEMP(I)
      DO 2200 J=1,3
 2200 BRAILL(J,I) = BTEMP(J,I)
      END


      SUBROUTINE STAKTB
C     SET POINTERS TO INTRODUCE A NEW ROW IN THE CURRENT STACK
      IMPLICIT INTEGER(A-Z)
      COMMON /STAX/ TS,HS,CS,PS,TAILS(2),HEADS(2),CURRS(2),CURTYP,STKIND
     *,HFSTAK,ELT(19,30),SPCONT(19),NOELTS(19),PVSTAK(19),NXSTAK(19)
C
      IF (HFSTAK .EQ. 0) RETURN
      N = HFSTAK
      HFSTAK = NXSTAK(N)
      NXSTAK(N) = 0
      PVSTAK(N) = TS
      NXSTAK(TS) = N
      TAILS(CURTYP) = N
      TS = N
      CURRS(CURTYP) = N
      CS = N
      IF (STKIND .EQ. 5) STKIND = 2
      END


      SUBROUTINE PCLEAR
C     RESET PROOF OUTPUT BUFFER TO BLANKS
      IMPLICIT INTEGER(A-Z)
      COMMON /OWAREA/ BRAILL(3,40),PROOF(40),CODE(40)
      COMMON /SPAC/ SPACE
```

```
C
      DØ 200 I=1,40
      CØDE(I) = 0
      PRØØF(I) = SPACE
      DØ 200 J=1,3
  200 BRAILL(J,I) = SPACE
      END


      SUBRØUTINE RESET
      IMPLICIT INTEGER(A-Z)
      CØMMØN /STV/ STVAR(10),NSV
      CØMMØN /TTL/ TTLENG
      CØMMØN /N/ LETN
      CØMMØN /INDENT/ MARGIN,INHEAD
      CØMMØN /CØMP/ CØMPUT
      CØMMØN /DUB/ DØUBLE
      LØGICAL DØUBLE
      LØGICAL INHEAD,CØMPUT
C
      TTLENG = 0
      MARGIN = 0
      DØUBLE = .FALSE.
      CØMPUT = .FALSE.
      INHEAD = .FALSE.
      DØ 100 I=1,NSV
  100 STVAR(I) = LETN
      END


      SUBRØUTINE DECØDE(N,Y)
C     DECØDES FIRST N CHARACTERS ØF BUFFER TØ GIVE INTEGER IN Y
      IMPLICIT INTEGER(A-Z)
      CØMMØN /BUFF/ BUFFER(10)
C
      Y = 0
      DØ 200 I=1,N
  200 Y = 10 * Y + IAND(15,ICH(BUFFER,4*I-3))
      END
```

APPENDIX 2

EVALUATION

In order to evaluate the system a small-scale pilot service was operated for producing single copies in braille of short documents for blind subjects. The subjects were chosen from those who approached Warwick Research Unit for the Blind asking for documents to be transcribed into braille. In the first few months of operation the system was undergoing almost continual modification based on informal feedback from the blind subjects which made it impractical to start any formal evaluation programme.

Examples of experimental material produced includes:

Domestic

- Washing machine instructions
- Blender/Grinder - instructions and recipes
- Refrigerator instructions
- Automatic dishwasher instructions
- Recipes
- Diabetic recipes
- Electric cooker instructions
- Instructions for record player
- Instructions for Dritz bound button hole maker
- Russell Hobbs coffee percolater
- Sanyo Stereo Set
- Teletron Cassette Tape Recorder/Radio
- Habitat Chicken Brick
- Sinclair 5-function calculator
- Information on freezing food.

Leisure

- Record cover - classical
- Times Literary Article

- Knitting patterns
- List of holiday bungalows
- Poetry/songs
- "Coffee Pot" activities
- Royal Northern College of Music - Programme
- Arts Centre, University of Warwick - Programme
- Leaflet on hotels
- Record sleeves
- List of monthly record releases

Religious

- Christmas Carols for vicar
- Methodist Church Services
- Catholic Mass Sheet
- Methodist Church - Aylesbury circuit plan

Education

- Song sheets
- Sociology "O" level handouts
- English handouts
- Lesson notes - primary school
- Examination papers (old)
- Bibliographies
- City & Guilds Certificate in Further Education - Information
- TUC Training College handouts

Employment

- Non-accidental injury pamphlet - social worker
- Open University material
- ATTI leaflet
- University library - information
- Agendas
- Minutes
- APEX handout / Musician's Union handout
- Letters
- Dialling Codes
- Reports
- Bibliographies
- NASWB
- International Dialling Codes

Miscellaneous

- Minutes and agendas of voluntary organisations
- Bus Timetables
- Train Timetables
- Mortgage Statement
- BBC Radio Birmingham
- Disabled Parking Stickers - Information
- Children's stories
- Call Charges
- Local dialling codes
- Newspaper articles
- Birth pill instructions

The proportion of the experimental material in each group is:

Subject	%
Domestic	10.4
Leisure	15.4
Religious	2.5
Education	25.4
Employment	29.1
Miscellaneous	17.2

Errors can be caused by typing errors in the text, incorrect control characters, incorrect choice of contraction by the computer program, malfunction of the embosser or damage to the braille in transit. Little is gained by just measuring the number of miscontractions caused by the translation program unless one also measures the error rates for the various systems using manual transcribers. Once the error rate is relatively low, it becomes a very expensive operation to attempt to measure it accurately.

A practical measure of the error rate is the number of users who find the number and types of errors unacceptable for their application. For instance any error in a list of telephone numbers is serious, but a miscontraction may often go unnoticed.

A braille questionnaire was circulated to subjects who had used material generated by the most recent system. The results are (N=23):

1. Mean age = 40.3 years (σ = 12.6 years)

2. Occupations

Professional	9
Non-professional white-collar	7
Students	2
Manual workers	1
Not employed	4

"The following 4 questions use a 1 to 5 scale for answering. For example 1 is very poor, 2 poor, 3 average, 4 good and 5 very good. You should just write down the number which best describes your own opinion. In these questions I have just given you the end points on the scale although you can answer with any number between 1 and 5".

		mean	σ
3.	For your applications, the turnround time is: (1 is so slow as to make the service useless and 5 is perfectly acceptable).	3.8	0.9
4.	Are the miscontractions caused by the computer program: (1 is so bad as to stop you using the service and 5 is perfectly acceptable).	4.6	0.6
5.	Is the physical quality of the braille: (1 is very poor and 5 is good).	4.3	0.9
6.	Is single-sided, as compared with double-sided, embossing: (1 is so severe a disadvantage as to stop you using the service and 5 is perfectly acceptable).	4.9	0.3

7. If there was a central document transcription service, or a number of regional ones, how much would you use it on average (braille pages per month):

1. less than 10
2. 10 to 50
3. 50 to 100
4. 100 to 1000
5. more than 1000

mean = 2.5
σ = 0.9.

8. Would you want the document service to transcribe specialist braille codes such as music, mathematics and computing? If so, which ones?

Code	Number
Music	6
Mathematics	1
Computing	1

9. How many embossed diagrams would you want per month (on average):

1. none
2. less than 1
3. 1 to 100
4. 10 to 100
5. more than 100

Mean = 1.4
σ = 0.6.

APPENDIX 3

FURTHER RELATED RESEARCH PROJECTS

This appendix lists some of the areas in which further substantial research may be expected to yield significant results of value to the ultimate user. Certain items are already being investigated, and reference should be made to the International Register of Research on Blindness and Visual Impairment (Gill, 1975) for details. This list should not be taken as implying that the short document service should be suspended or that the prime object of this study has been achieved. Rather it should be regarded as an indication of the continuing program envisaged as a result of discussions, and comments and experience from some four years of work in this area and from the particular information gained whilst undertaking the project described in the main body of this report.

1. Motivation in braille learning

Time has not permitted an investigation on whether the provision of material of particular interest to an individual can help motivate him to learn braille. This study would necessitate particular assistance from other experts, notably in the field of education.

2. Formal definition of braille

A formal definition of contracted braille would facilitate the writing of translation programs. A considerable literature exists on the formal definition of natural languages and has assisted the machine translation of one language to another (e.g. Russian to English). No such complex analysis has been undertaken for translation into braille, where similar advantages may result. It may be that successful completion of this formal definition would ease the production of translation of many other natural languages to braille.

3. Mathematics

The input processor and DOTSYS should be developed in order to handle the English mathematics braille code.

4. Choral Music

Design of an interactive system for the input and translation of simple choral music by DOTSYS. Unlike existing systems and systems under development, computer based editing is envisaged as a vital element of this proposal.

5. Tables

Design of an interactive system to assist the operator in the layout of tables when using DOTSYS.

6. New page control

The operator should have an extra instruction for DOTSYS which checks the amount of space left on the page to ensure that a table is not unnecessarily split between two pages.

7. Fill-in character

For tables it would be sometimes useful to replace blanks by a fill-in character such as a dash. DOTSYS could be modified to do this automatically.

8. Alternate page inversion

DOTSYS should be modified to include a post-processor to handle alternate page inversion which could make significant financial savings, by minimising the amount of manual binding work required.

9. Financial information

With the increasing use of digital systems by financial organisations, more financial information could be produced automatically in braille, e.g. bank statements. This does not usually require a pre-processor to a braille translation program such as DOTSYS since the quantity of text is often minimal, and much of it is drawn from a limited vocabulary of standard words.

10. Telephone directories

Many organisations store the data for their internal telephone directories in digital form. So it would be possible to automatically generate braille editions either alphabetically or by department.

11. Computer-based composing systems

Some printers are using computer-based systems for the generation and correction of text. These systems often incorporate features which are very useful for braille translation, e.g. different symbols for apostrophe and single quotation mark. With a suitable pre-processor, these tapes appear to be ideal for braille production requiring only that the operator specifies the layout for tables, which is best done interactively.

12. Design of data structures

Sometimes the braille producer can specify desirable features when a data structure is being formulated. For instance magazines such as New Beacon, are produced in inkprint as well as braille, so the use of one database for both editions could offer significant savings. It should be possible in many cases to completely eliminate the currently duplicated data preparation and proof reading.

13. Automatic indexing

A modification to DOTSYS that would be useful for reference books is a facility to allow the typist to flag terms required in the index and for DOTSYS to automatically generate the index.

14. Double indexing

When computer-generated compositor's tapes are available for books, some users would find it advantageous to have a double indexing system giving both inkprint and braille page numbers; this could be done automatically with the development of suitable software.

15. Current awareness services

Pre-processors need to be developed in order to selectively produce braille listings from the standard databases such as INSPEC. It should be possible to distribute the individual braille listings no later than the publication of the inkprint abstract journal.

16. Retrospective abstract services

It would also be technically possible to offer a retrospective service using these same databases. However the choice of index terms becomes more critical so an information broker may be necessary.

17. Small-configuration DOTSYS

It would significantly affect the economics of computerised braille production in both the developed and developing countries if DOTSYS were modified to run on a minicomputer or a microprocessor.

18. Compiler-compiler

It has been proposed that a braille translation program, based on the compiler-compiler principle, could be most beneficial particularly for facilitating a language-independent translator.

19. Design of tables

Simple tables can be translated into braille automatically but this is not true for complex tabular presentations. For instance it has been found preferable to use an inverted file structure when translating a scan-column index into braille. The design of tables is a large area deserving of research effort.

20. Design of embossed diagrams

So little research has been done on the optimum design of embossed diagrams that it is impossible even to list all that still has to be done.

21. Double-sided embosser

The development of a digitally-controlled high-speed double-sided braille embosser will soon become a matter of urgency.

22. Stereotype machine

There will also be an increasing demand for a high-speed digitally-controlled braille stereotype machine. Since the potential market is small and the development costs high, it will have to be undertaken as an international project with guaranteed sales.

APPENDIX 4

"SOME DEVELOPMENTS IN COMPUTER-AIDED INFORMATION SERVICES FOR THE BLIND".

by Professor J.L. Douce

Chairman's address to the Control and Automation Division of the Institution of Electrical Engineers on 7th October 1975.

To be published in the Proceedings of the Institution of Electrical Engineers.

SOME DEVELOPMENTS IN COMPUTER-AIDED INFORMATION SERVICES FOR THE BLIND

John L. Douce, DSc FIEE

Abstract

This paper reviews the needs of the blind community for printed material, and surveys current developments in the applications of automation to increase the availability of braille. The problems of automatic translation of English text to braille are considered.

Some possible systems for giving a greatly improved access to braille are surveyed and an automated system for map and diagram production is reviewed.

The future provision of information services for the blind, using special systems or existing systems with special terminals, is discussed. The technical problems are not trivial but collaborative ventures which, with international cooperation, could offer solutions in a relatively short time-scale.

* - * - * - *

Professor Douce is Professor of Electrical Science at the University of Warwick and Director of the Inter-University Institute of Engineering Control.

INTRODUCTION

One of the most significant handicaps resulting from blindness is the severely restricted access to information of many kinds particularly to that conveyed to sighted people by the printed text and diagram and the medium of television. At the present time, significant developments are occurring in the application of automation to augment the supply and accessibility of this type of information needed by the blind for the purposes of employment and leisure activities. This paper reviews some of the current problems with regard to the provision of textual and diagramatic material, and examines possible future systems and their potential applications.

1. THE BLIND POPULATION

A person may be registered as blind in the U.K. if he is unable to read at a distance of three feet an optical test card which can be read with normal vision at a distance of sixty feet. It is perhaps more meaningful for the purposes of this discussion, to consider the term 'blind' as applying to that section of the community who find conventional ink-print unacceptably difficult to read, and for whom braille or similar embossed material offers the only useful printed reading material. Perhaps the impact of the tape recorder has been overestimated as an information storage device, due to the lack of indexing capabilities and the problems of handling numerical information.

Statistics concerning the visually handicapped section of the population cannot be presented with the desirable accuracy, since the definitions of 'blind' and 'partially sighted' are not particularly closely related to the group involved, and it is believed that many people who are eligible for registration do not register for a variety of reasons. However, appreciation of the size and age distribution of the population may be obtained from Table 1 and Figure 1, which show the age distribution of the 106,000 blind persons registered in England and Wales in 1974.

From this table, and noting also that over 60% of the newly registered blind are aged 75 or over, age alone is likely to result in the majority of the group not learning braille. Furthermore, almost 50% of the children born blind suffer additional handicaps. The overall result is that only about 10% of the registered blind read braille regularly and competently. Thus it must be appreciated that even a non-specialised aid such as braille cannot be expected to aid the majority of the blind.

Table 1. Total blind registrations on 31st March 1974 in England and Wales.

Age Group	Male	Female
under 21 yrs	1783	1419
21 - 39	3177	2252
40 - 59	7378	6071
60 - 74	11986	15543
over 75 yrs	15981	39884

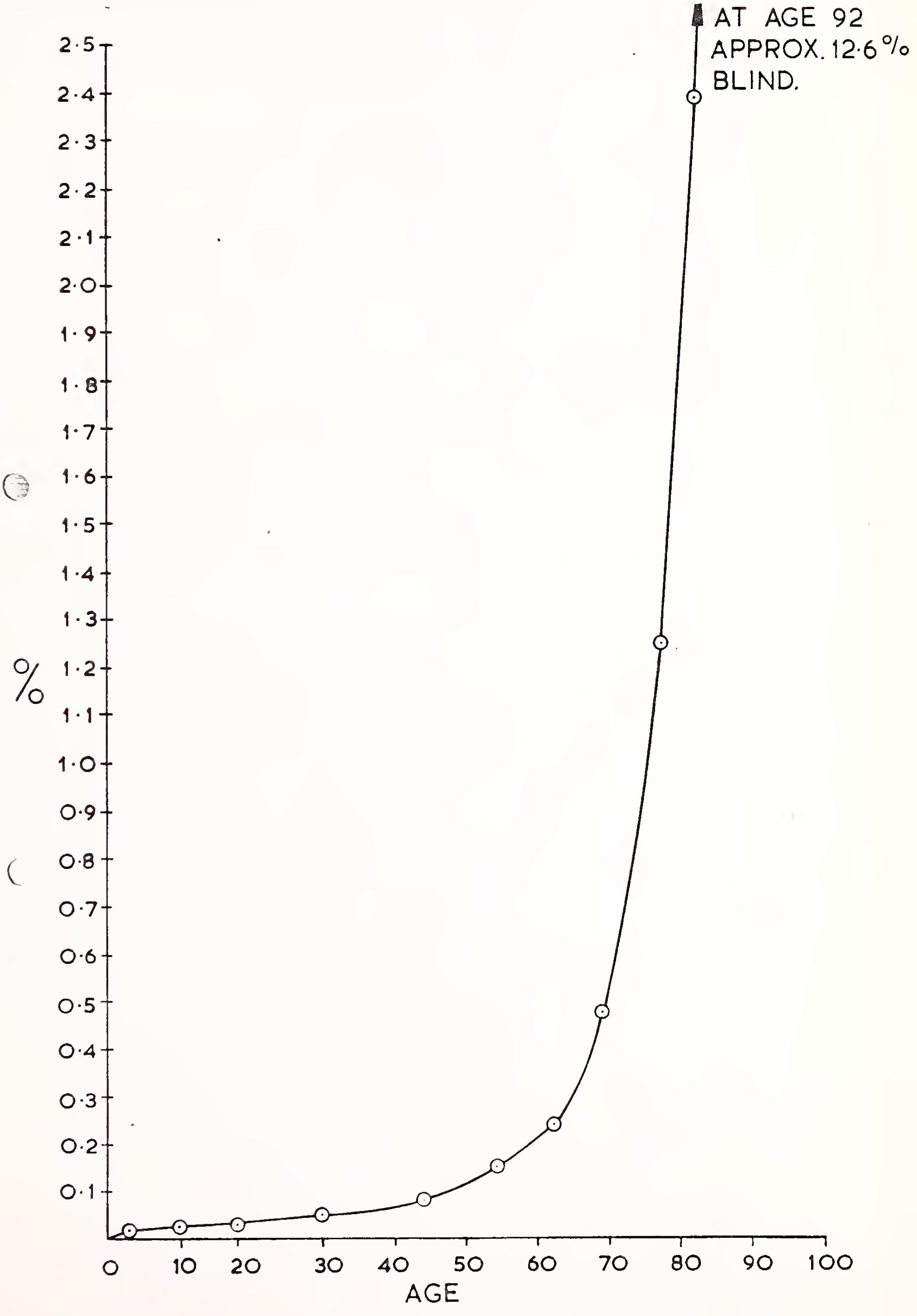


Fig 1. % of each age group (England and Wales), for 1974 of registered blind

When the blind population is examined by professional employment, we find only a tiny fraction of the population in this category, and even the most accessible occupations, such as physiotherapy and teaching have only of the order of one hundred people in each group. The traditional approach has been to find jobs which do not require vision, rather than developing aids to enable a blind person to do the job of his choice.

Thus it may be concluded that the demand for specialised aids confined to one age group or one professional occupation is small. The most useful devices and systems are likely to be versatile and able to assist relatively large numbers of users at an acceptable cost.

This is not to minimise the usefulness of many aids currently available, often developed by talented workers with much voluntary effort and expense. It is rather to attempt to devote effort where the highest cost-effectiveness may be anticipated.

At present, limited information access seems to be the common theme of many frustrations and disadvantages of blindness. With the increasing use of digital data bases and distributed computer terminals, the situation could easily deteriorate in the near future in the absence of a directed program of research and development.

It seems inevitable that braille will be used in increasing amounts in the future, in spite of its deficiencies, and it is appropriate to consider the achievements and possible developments automation offers in this area in some detail.

2. BRAILLE

The most widely used medium for printed material, including books of text, mathematics, music and personal notes and letters is based on the braille cell, developed one hundred and fifty years ago by the Frenchman Louis Braille. It is remarkable how little the fundamental information element has changed since its origination, and the basic design seems close to achieving maximum information density.

The braille cell consists of six dot positions, Figure 2, spaced 2.5mm apart, with adjacent cells spaced 4mm apart.

Figure 2.

Information is conveyed by raising certain dot positions, so that in principle, 64 different patterns are available. These cells are packed at about 1.6 per sq. cm., which should be compared with ink-print text (for which a greater character set is available) with a packing density of the order of 22 characters per square cm. The ratio of these two figures, in conjunction with the thick paper necessary for braille text explains the weight, volume and to some extent the cost of the library of a blind professional person. For example the bible, which contains 773,746 words, takes up 72 braille volumes but NCR have a sighted version on a 50 mm square microform. At present, there is no braille equivalent of high density storage media although the need is obvious.

Grade I braille uses one cell pattern for each letter of the alphabet, with some of the spare patterns for simple punctuation signs and abbreviations of a few very common words. Thus it is a trivial matter to automate the translation of text to Grade I braille since a small look-up table is all that is required.

The automatic translation of text to English (or American) Grade II braille, rather than Grade I, presents formidable difficulties if perfection is desired. To appreicate the nature of the difficulties, some illustrative aspects are worth remarking. Following standard practice, groups of letters represented by a group of one or more braille signs are printed in capital letters, with an oblique stroke inserted where necessary for clarity to separate adjacent contractions written with no separating space.

Grade II braille makes extensive use of contractions to reduce the length of text, with a resulting reduction in the number of pages of braille and, for skilled users, in the reading (and writing) time.

One simple and obvious step is to use the letters of the alphabet to represent whole words, and 23 letters are thus employed. Since the alphabet requires only 26 of the 63 available braille signs there are 37 signs available for common words (e.g. AND/FOR/OF/THE/WITH), punctuation, and for contractions of common groups of letters (e.g. CH/WH/COM/DIS/ING).

Many of these rules are simple to formulate and hence to program, but difficulties of various magnitudes soon become apparent.

For example, some contractions may only be used at the beginning of a word or at the beginning of a braille line. This means that the format of the output print must be considered during the translation phase. More difficult, in automatic translation, some contractions can only be used when they represent a complete syllable - for example DIS may be used in DISTURB but not in diSHes. Other contractions may only be used where the pronunciation is unchanged. Thus UNDER may be used in blUNDER but not in laundER. On the other hand ONE may be used as a contraction when all the three letters it represents are pronounced as a single syllable - ST/ONEs but not anemone. Some, but not all, braille abbreviations may only be written within longer words where the sense of the contracted part is

unchanged. Thus the abbreviation for 'after' may not be used in 'rafter' nor that for 'across' used in 'lacrosse'. Even the general rule for dividing words at the end of a line, 'divide between syllables', is fraught with difficulty for the mechanised translator.

In spite of these difficulties, the savings achieved with Grade II braille, resulting from the reduction of about 40% in the number of cells required, has encouraged many groups to program the conversion. Perhaps surprisingly, at the current time, cost saving, projected or actual, is not the prime motivation for automation. The main reason is at present the lack of skilled braillists to meet a greatly increasing demand.

The possibility of automatically translating text to contracted braille has attracted considerable attention in the last decade. The developers of these systems foresee many potential advantages, such as consistent results achievable by automatic translation and fast turnround, but most of these are only now being investigated at the pilot study stage.

Two types of computer program have been developed, the dictionary-based conversion in which of the order of 80,000 common words are stored, and smaller programs which try to translate by systematic application of the ordered rules and exceptions, and exceptions to the exceptions, with a relatively modest dictionary. Most programs are in high-level languages such as FORTRAN, COBOL or PL1, and modest efforts at international collaboration have been undertaken. It should be noted, however, that Grade II braille is not an international standard - even the conversion of an American to English grade II translation program is not trivial. On a medium-sized machine available at Warwick, a translation speed of about 5000 words per minute is currently being achieved, using 13k words of store.

There has been pressure from diverse groups for changes in the rules or semantics of grade II braille, and the increased interest in automated

translation is giving added impetus to this minor revolution. Other countries have initiated this change, apparently successfully. The main motivation for changing some of the rules of grade II braille is to make it easier to learn by blind readers and sighted transcribers.

For example it may be greatly regretted that the very valuable single letter k is devoted to the relatively infrequent work KNOWLEDGE, and a simplification of the use of contractions could well aid both the user and the translator. The revision of braille however is a surprisingly emotional subject and has led in the past to bitter argument and to the permanent rupture of close friendships. It must also be borne in mind that a great amount of braille is still provided by a voluntary labour force built up over many years, and the introduction of a new system with scientifically chosen and evaluated contractions would introduce major transitional problems over a long period.

Two working systems for text translation are illustrated diagramatically in Figure 3. The first is used at the R.N.I.B. for book production, the final output of the system being embossed zinc plates for use in a printing press for producing two sided braille sheets, and considerable effort is devoted to ensuring that the end product is as accurate as possible. The contrasting system at Warwick provides a document service aimed at rapid and cheap production of short documents in small production runs, the error rate being relatively higher but hopefully tolerable, and an acceptable alternative to increased cost. Reduced human intervention coupled with good operator-machine dialogue seems to be the only way in which translation costs can be substantially reduced, and the amount of braille increased, perhaps tenfold, in the near future.

The provision of braille text is only one part of the problem of information provision for the blind. Much more difficult is the presentation of tabular information, e.g. timetables, mathematics and music scores.

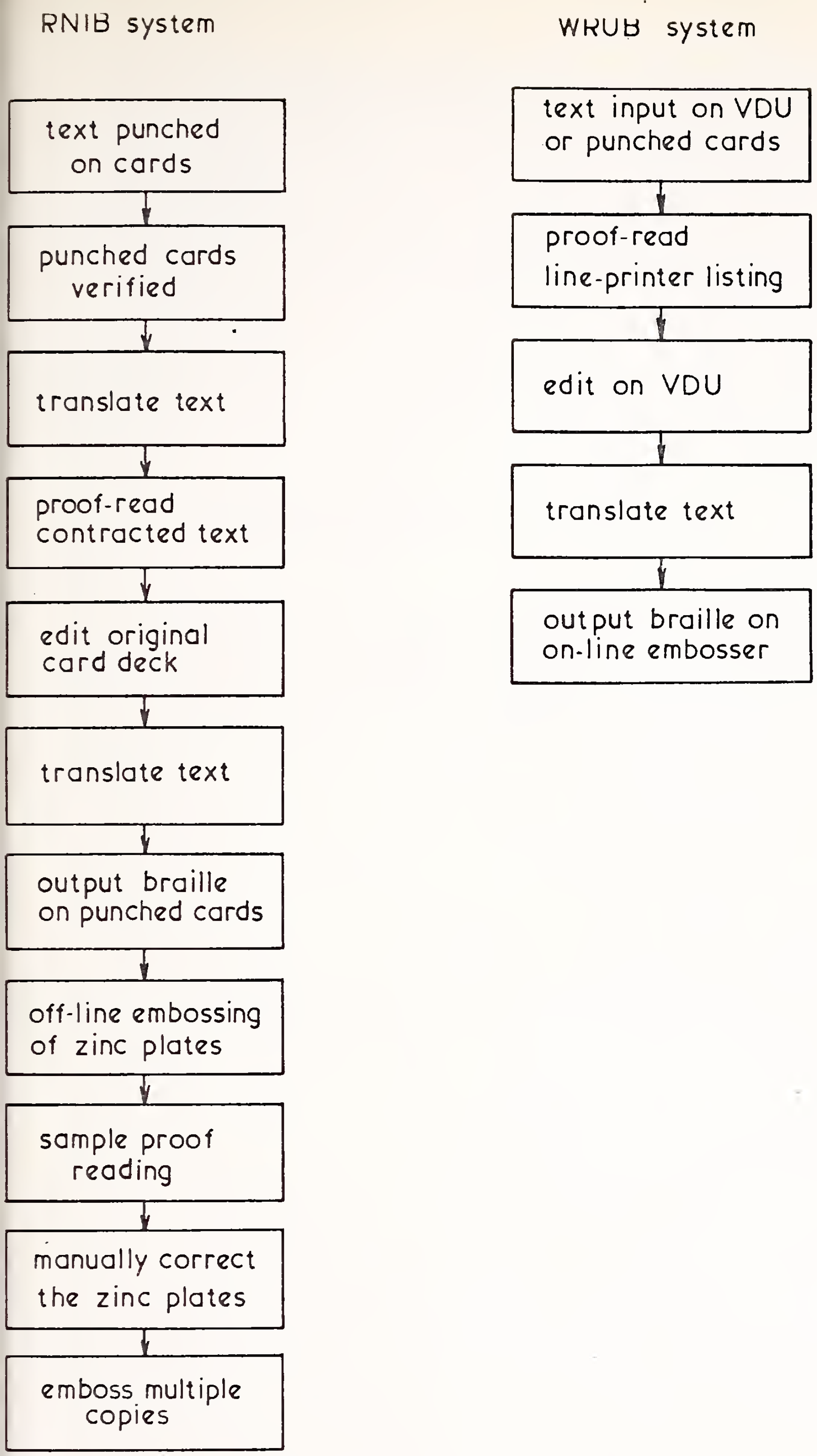

RNIB system
text punched on cards
punched cards verified
translate text
proof-read contracted text
edit original card deck
translate text
output braille on punched cards
off-line embossing of zinc plates
sample proof reading
manually correct the zinc plates
emboss multiple copies
WRUB system
text input on VDU or punched cards
proof-read line-printer listing
edit on VDU
translate text
output braille on on-line embosser

Fig. 3. The main steps in the automated production of contracted braille.

The American Printing House for the Blind is making a serious attack on this latter problem, with a music typewriter connected to a card punch for initial data input (Figure 4). The translation program for single stave music will be at least four times larger than that for text translation.

3. MAPS AND DIAGRAMS

The sighted scientific community regards graphical presentation of information as a vital ingredient of information storage and communication. A well designed diagram can be scanned rapidly for extraction of the salient features, and a small area selected almost instantaneously for detailed examination. The information density on a complex figure, with multicolour printing can be extremely high.

Graphical information for the blind is usually in the form of embossed sheets, read by touch. The limited resolution of the fingertips, the general lack of properly designed symbols, and the difficulty in scanning the diagram, severely restrict the amount of information presented.

In an attempt to assist the proper design of tactual figures, research effort has been directed to the design of symbols for points, lines and areas, the objective being sets of symbols which are discriminable by the majority of readers. The one advantage of embossed symbols over ink-print is the added flexibility available in that height can be used as an additional coding dimension. This has led, in particular, to the use of a line with height varying in a saw-tooth fashion along its length. This line is extremely acceptable for conveying directional information, to show information flow patterns on block diagrams or the sense of traffic

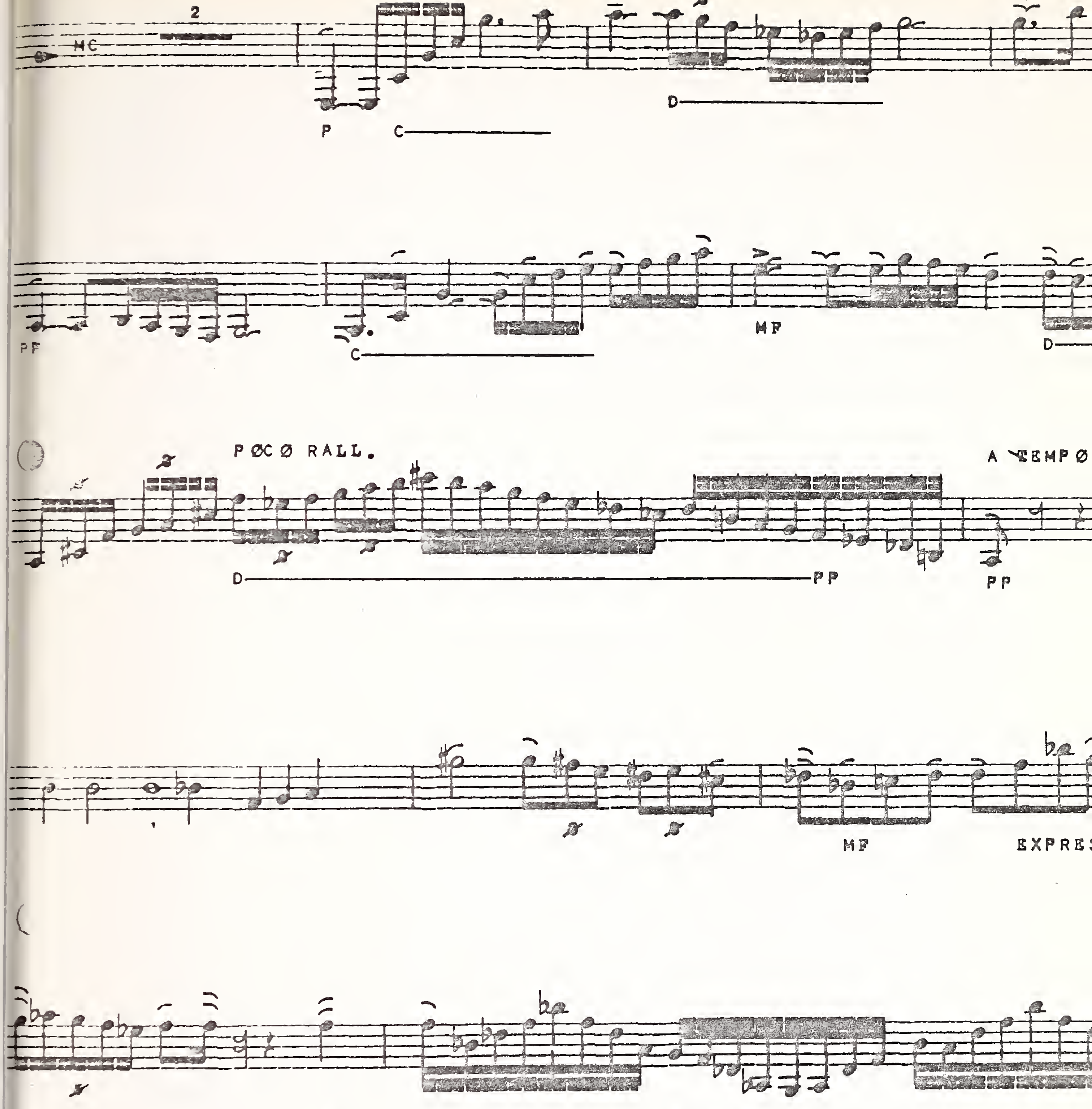


Fig.4 Printout of a digitised music score from the system under development at the American Printing House for the Blind.

flow in a one-way street, and tests prove this representation to be superior to the embossed version of the traditional arrowhead marking.

The initial research on symbols was a joint project involving the Engineering Department at Warwick and the Blind Mobility Research Unit at Nottingham University. One result of the studies has been the production of a kit of symbols for amateur map makers, useful both to provide proper symbols to other constructors and also hopefully to help standardise on a selected notation.

The Warwick production facilities can now make maps of a high quality at relatively low cost, and several dozen orientation maps have been provided for local and national facilities. As with automated braille production, the main cost is the input of the information to the system, which requires considerable design skill.

The system implemented at Warwick University is a fairly integrated computer-based production facility, offering good interactive facilities to the designer with the aim of providing relatively cheap high quality maps and diagrams in small production runs. The designer must have an annotated master diagram, using his skill to select the essential features without introducing an unacceptable quantity of information.

Data from the visual copy is supplied to the system via either a graphics terminal with joystick or through a simple coordinate table. Control commands specify the type of line - solid, dotted, dashed, sawtooth, etc. and the height of the line. Special symbols to specify landmarks or hazards can be inserted, and these symbols can be chosen from the standard set or be user-designed.

Versatile editing facilities enable the graphical data to be modified by rescaling the diagram, changing line parameter, deletion and substitution of data and by squaring up of lines which are 'sufficiently' close to being perpendicular. Text can be positioned where desired, and automatic translation to grade I braille is provided. The final data can be stored on paper tape for reference and later modification if necessary. The final output of the system is an engraving in laminated plastic produced by a numerically-controlled machine tool, This is driven by the computer used in the design process or by a smaller computer using paper tape control data. A male epoxy resin master is cast from this female engraving, and used to manufacture the required number of copies by conventional vacuum forming.

Several hundred maps of Euston Station, reproduced in Figure 5, with the alphabet replacing the corresponding braille symbols, have been distributed. This particular location has, in the past, proven particularly difficult to transverse, with no distinguishing floor textures nor acoustic cues to aid the location of central islands of importance like entrances to the taxi stand or to the underground. One finding from the questionnaire, filled in by the users of this map, was that they all found that the map greatly increased their knowledge of the environment and facilities at Euston Station.

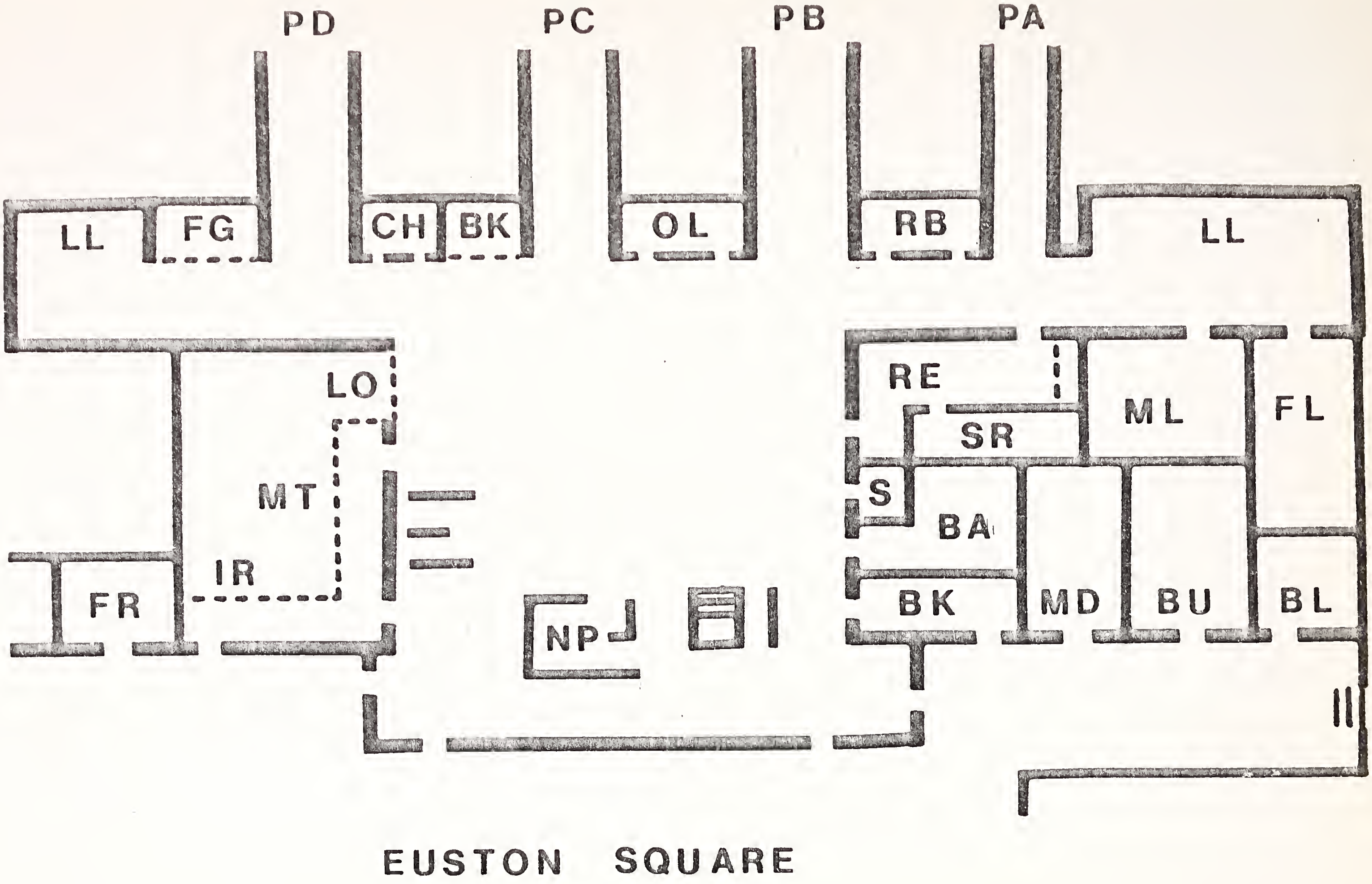


Fig.5 Sighted Version of braille map of Euston Station.

4. SOME CURRENT DEVELOPMENTS

The most immediate advance in the provision of information due to automation will probably be the computer-aided production of braille books and periodicals. It is not difficult to envisage a ten-fold increase in the output of braille, using existing techniques, and gradually reducing the amount of manual checking and editing, with better output facilities by large-scale introduction of on-line braille embossers.

One device of great potential is the braille writer under development by Dr. A. Grunwald in Chicago. This displays a line of braille on an embossed plastic loop or belt, the information being erased after presentation prior to re-embossing with subsequent information. Text is stored in digital form on magnetic tape, with some indexing capability, this medium considerably reducing the volume and cost of a braille library. The speed of the plastic belt is controlled by the user, and this feature, together with the continuous rolling display of braille, appears to provide a useful increase in the rate at which braille can be read.

It is also hoped that more integrated information services will become of proven reliability and more widespread in the future. A possible system, the ARTS (Audio-Response Time-Sharing) Service has already been demonstrated, in which many users interrogate a central data bank with typewriter input and synthetic speech feedback and optionally printed braille output, either locally, on-line, or by central print-out. Some combination of these two previous systems could well offer a braille computer terminal as a direct alternative to a visual display device, with considerable possibilities for increasing the employment opportunities for the blind.

More mundane perhaps, but by no means trivial, is the increasing use of digital compositors tapes for the source material for braille. Although

this appears, at first consideration, to offer an error-free input, the practical problems are considerable, since there is a daunting multiplicity of codes amongst printers and final proof correction may never in actual useage produce a clean tape, and control characters for ink print and braille version present a challenging problem.

However, more promising are the computer-based systems currently being introduced by some printers, where the text is corrected on a visual display unit and the error-free data can be output on magnetic tape. Some of these systems have the advantage that they include flags indicating headings, page numbers, italics and footnotes, and they also differentiate between apostrophe and single quotation mark which are represented by different symbols in braille.

With the increasing availability of data in machine-readable form, many possibilities exist for completely automatic braille production, with the consequent reduction in routine human intervention and hence in cost. For example at Warwick a pilot scheme for the translation of bank statements to braille is now being undertaken on a routine basis, with a twenty-four hour turnround from receipt of the magnetic tape containing the data to print out both inkprint in line printer output and braille. These statements are complete, apart from deletion of the account holders identification. Internal telephone directories are fairly readily produced in braille from digital data in standard format, with the incidental benefit that alphabetic lists, as well as the perhaps more usual listing by section or department, can be produced for switchboard operators if required.

Professional blind people have a particularly serious problem in keeping up to date in their fields. For instance it is particularly difficult for a blind person to scan the journals in a library. One attempt to alleviate this problem is being made by the Warwick Research Unit for the Blind in cooperation with the Institution. The pilot scheme produces

selective listings of the INSPEC Computer and Control Abstracts in braille. The feedback from the blind programmers participating in this experiment has been sufficiently encouraging that the possibility of extending the pilot scheme to other subjects is being investigated.

If these pilot schemes are successful, then it is hoped they will be taken over, in due course, by the established service organisations. At the present time these activities serve the important purpose of keeping research in close contact with the ultimate users, though financing such useful developments is incredibly difficult.

It is hoped that the provision of more braille will provide to increase the number of braille readers, by making more teaching material available and by a larger library giving more motivation for the better and wider use of braille.

CONCLUSIONS

Some of the possibilities for automation in the provision of information to the blind have been outlined. It is an area where interdisciplinary effort is absolutely essential, and international cooperation can offer unusual dividends. There is a long way to go before the blind are able to fully utilise their talents in employment and in their leisure activities.

Although this survey has concentrated on the aids which automation and relatively complex systems may offer, it is important to appreciate the equally important apparently trivial aids which are lacking for the blind. It is particularly regrettable that gadgets are frequently developed and never produced on a large scale and marketed, whilst other devices, like an embosser of braille onto adhesive plastic strip are suddently withdrawn. For instance, of all the gadgets developed in British universities in the last years, none are currently available, as standard items, to the British blind population.

A major advance could well result if the cost-effectiveness of complete systems could be properly measured. This assessment should include effects of increase and higher grade employment, reduced welfare costs and the value of a better awareness of the present day environment which greater information availability can offer to the blind.

ACKNOWLEDGEMENTS

I am indebted to Dr. John Gill who has been the principal worker at the University of Warwick in this area for the past five years. The late Dr. J.A. Leonard and his successor Dr. J.D. Armstrong of the Blind Mobility Research Unit have provided much useful collaboration. The work would have been impossible without the facilities provided by the Science Research Council, and the assistance of the Royal National Institute for the Blind, the American Foundation of the Blind, and the Medical Research Council for the early work. Many individuals have given considerable time and effort to guide the orientation of the research, in particular Dr. Fred Reid, of the National Federation of the Blind and Dr. M.J. Tobin of the Research Centre for the Education of the Visually Handicapped.

REFERENCES

"Braille mathematics notation". Royal National Institute for the Blind, London, 1973.

Coleman P.W.F. "Computer terminals for the blind". Electronics and Power, Vol. 18, March 1972; pp 84 - 86.

Douce J.L. & Gill J.M. "Computer drawn maps for the blind". Electronics and Power, Vol. 19, No. 14, August 1973, pp 331 - 332.

Gerhart W.R., Millen J.K. & Sullivan J.E. "DOTSYS III: a portable program for grade 2 braille translation". MITRE Corporation, U.S.A., 1971, 139 pp.

Gill J.M. "Orientation maps for the visually handicapped". British Cartographic Journal, Dec. 1974, pp 94 - 98.

Gill J.M. "Methods of increasing the accessibility of reading material by the blind". The Louis Braille Conference, Cambridge, England, Jan. 1975, pp 155 - 162.

Gill J.M. & Reid F. "Visual handicap: aids and support". Open University, The Handicapped Person in the Community, Unit 6, 1975.

Gill J.M. "Non-visual computer peripherals". American Foundation for the Blind, New York, 1975.

Gill J.M. "The international register of research on blindness and visual impairment". Warwick Research Unit for the Blind, University of Warwick, England, 1975.

Goldish L.H. "Braille in the United States: its production, distribution and use". I.R.I.S., American Foundation for the Blind, New York.

Grunwald A.P. "A braille reading machine". Research Bulletin of the American Foundation for the Blind, No. 16, pp 74 - 78.

Ingham K.R. "Audio-response time-shared service bureau for the blind". MIT Report, 1971.

Irwin R.B. "The war of the dots". American Foundation for the Blind, New York, 56 pp.

Lawes W.F. "The feasibility study into the usage of computers by and for the blind". Royal National Institute for the Blind, London, Jan. 1975.

Proceedings of the workshop "Towards the communality of algorithms among braille transcription systems for multilingual usage". University of Munster, Germany, March 1973.

Proceedings of the workshop "Computerised braille production". Danish Association of the Blind, Copenhagen, Sept. 1974, to be published.

"A restatement of the lay-out definitions and rules of the standard English braille system". Royal National Institute for the Blind, London, 1969 & 1971.

Schack A. & J. "Studies in the automation of braille mathematics and music". Final Report to the Department of Health, Education, and Welfare, American Printing House for the Blind, June 1969, pp 8 - 78.

Schiff W., Kaufer L. & Mosak S. "Informative tactile stimuli in the perception of direction". Perceptual and Motor Skills, Vol. 23 (Monogr. Suppl. 7), 1966.

APPENDIX 5

"METHODS OF INCREASING THE ACCESSIBILITY OF READING MATERIALS BY THE BLIND".

by Dr. J.M. Gill

Paper presented at the Louis Braille Conference, Cambridge, England, January 1975.

Methods of increasing the accessibility of reading materials by the blind

J.M. Gill

A totally blind person has three main methods for accessing written information.

1. Use of a reading aid with the information in ordinary sighted form. The reading aid can be of either direct or indirect access type; the former involves automatic character recognition whereas the latter involves the user in skilled interpretation of the output. An example of a direct access type reading aid is the Textobrail and an example of an indirect access type is the Optacon.

2. The information can be recorded verbally in analogue form on magnetic tape or disc. This system has the advantage that it is compatible with the sighted community. The main problems are those of indexing and speed of data transfer.

3. The information can be in a tactile form. The most usual coding systems are braille or Moon, which are often embossed directly on paper.

This paper is limited to considering the problems concerning braille.

The previous speaker has described a system for increasing braille production by increasing the output of a skilled braille transcriber. Another approach is to use relatively unskilled typists and to use a computer to translate the text to an approximation to grade II braille. Since the demand for braille is greatly in excess of supply, it is essential to use all systems available to increase braille production.

The computer-based systems involve:

1. Input of text
2. Proof reading
3. Editing
4. Translation
5. Output

A basic system might be:

1. Input on punched cards or paper tape.
2. Listing on a line-printer for proof reading.
3. Editing on a visual display unit or even by correcting the punched cards.
4. Translation by a program such as DOTSYS III.
5. Output to an on-line embosser.

The advantages of such systems are that they are fast and do not require the use of highly skilled braille transcribers. Such systems are usually cheaper than employing manual transcribers.

There are applications where output in grade I braille is required. For instance a braille learner might want a list of football clubs playing the coming Saturday or the diary of events for the local darts club; however if this motivates the person to learn braille then it is worthwhile.

Input

Format information has to be included at this or the editing stage. Control characters are needed for paragraphs, forced new lines, running titles, centred headings and non-standard characters such as Greek letters. It may also be desirable to flag italics, foreign words and phrases, inflected letters and single letters other A,I or O. If footnotes are to be included in the main text, it will be necessary to flag the beginning and end of this text.

Tables

The major problem is material whose layout is an intrinsic part of the content, such as tables. The problem of tabular presentations is of the same order of magnitude as that of producing meaningful embossed diagrams. It is impossible to handle tables automatically in any but the simplest cases. However the problem of layout of tables and diagrams can be alleviated by using an interactive method on a visual display unit.

An example of this problem with tables is the O'Connor scan-column index; it is impossible to transcribe this into braille such that the user

can do a multi-column search.

Special codes

Further problems arise when inputting mathematics or music from an ordinary keyboard although feedback can be provided. For instance music can be coded using alphanumerics which can be directly displayed at the top of a visual display unit screen; underneath the music can be displayed in staff notation for checking against the original score. The data can be edited and then the braille can be displayed across the bottom of the screen.

Other modes of input

One method to reduce the cost of data preparation is to use prisoners. The text is input on a standard compositors keyboard with the output on punched paper tape. The trade unions should not object to such an arrangement since it is not in competition with open industry. The prison authorities might view the system favourably since the prisoners would be partly trained for employment in the printing industry on their release from prison.

Computer-compatible compositors tapes are possible input media but printers often make corrections on the type itself. Another problem is the variety of codes used by printers in this country but this is not insuperable since the pre-processor can be table driven.

However many organisations are storing digitally information which requires frequent updating such as internal telephone directories and bibliographies. Access to data bases such as ERIC, MEDLARS and INSPEC is very important to some professional blind people.

Optical character recognition is another mode of input but the accuracy and the range of acceptable typefaces are a function of bandwidth and therefore cost. At present OCR is not economically viable for this application.

Indexing

A sighted person with an inkprint textbook may use a number of different ways for locating a specific item:

1. The subject index at the back of the book.
2. Chapter titles, then serially through the chapter.
3. Subject headings which are repeated at the top of each page.
4. The reader may remember that it was "near the front of the book and mid-way down the page".
5. The reader may look for a diagram which he remembers is physically near the item he requires.

There are obviously other methods but the important aspect is that the system, or combination of systems, used is chosen by the individual reader although the choice will depend on what has been provided by the publisher.

The provision of indexing facilities for non-visual information systems deserves attention since it is an area which has tended to be neglected in the design of transcription systems.

Translation

A variety of computer programs have been written to produce an approximation to grade II braille. These programs will translate up to 25,000 words per minute although speeds of 1000 to 4000 words per minute are more common.

If there was a rigorous mathematically unique definition for grade II braille, the program could work directly from this definition. It is possible to consider translation to a grade II, which is not context dependent, being done on a microprocessor. This may be particularly relevant to developing countries since the cost of a microprocessor and the necessary peripherals may be less than the cost of training and using manual transcribers.

Outputs

If a large number of copies are required, the output can be off-line to machines which emboss the conventional zinc plates. Very often less than three copies are required so it is economical to emboss directly onto paper.

It is usual to interface the translation program and the output device with a post-processor which handles tasks such as page numbering. It can also handle alternate page inversion which involves embossing upside-down alternate pages of fan-fold paper in order to simplify the task of collating and binding.

Because of the considerable bulk of embossed braille and the increasing cost of paper, many alternative modes of storage have been proposed and sometimes developed to prototype stage. The storage media has usually been digital cassette, floppy disc, printed page or microfiche. All these systems involve the user in having a special reading machine. However the main application for such systems is for white-collar workers such as computer programmers where the text for manuals is sometimes available in digital form.

For example braille can be stored on microfiche. A device for the computer output of microfiche can produce 1 sheet per minute for a materials cost of 10 pence per sheet. Copies cost 2 pence. Each sheet can contain up to 100,000 braille characters.

A simple reading machine could consist of a line of photocells directly connected to a line of braille display. For all these digital storage systems it is possible to build more sophisticated reading machines which incorporate automatic search functions.

These methods for storing braille will not replace embossing on paper but may serve as an additional aid for some of the visually

handicapped in professional employment. Both digital cassettes and floppy discs can be used for writing as well as reading; a pair of storage devices is required for editing.

The main problem is in the design of the display. Although many displays have been built, little is known concerning the fundamental parameters of the man-machine interface.

Accuracy of computer transcription systems

Errors come from:

1. Incorrect input data which is not noticed at the proof-reading stage.
2. Incorrect or non-optimum format commands.
3. Incorrect choice of contraction by the translation program.
4. Malfunction of the output device.

The standards for accuracy should not be absolute but should be relative to the accuracy of other transcription systems in common use. Also the severity of the error should be taken into account; for example an incorrect choice of contraction may not significantly reduce the reading speed. The reading speed may not be impaired at all once the reader is used to a particular dialect of grade II braille.

Future research

In conclusion, there are eight areas which seem most deserving of attention by research workers.

1. Since data preparation is time-consuming and therefore expensive it is necessary to make best use of compositors tapes, other texts already held in digital form and other sources of inexpensive data preparation such as prisons.
2. The professional blind person is likely to find it increasingly important to have selective access to large data bases such as INSPEC. Some of the users may require on-line access through a soft-copy terminal in order to work in an interactive mode.

3. The development of interactive programs to assist in the design of tables and diagrams. This will require some psychological research on how tables and diagrams are 'read'.

4. The development of computer programs to handle specialist codes such as mathematics and music.

5. Fundamental studies on indexing systems.

6. Formal definition of grade II braille.

7. Design of inexpensive production systems for the developing countries.

8. Fundamental studies on the design of non-visual displays.

JMG/lcw

8.1.75

uments
er commands
a paragraph.
and any
ion, on a

otherwise

rst column

y do occur
gle spaces.

$HDE after. This
ntres one-line headings.
d for headings up to 36
ere a heading consists
6 characters the heading
as a paragraph ($P).

inning.

full stops in, e.g. R.N.I.B.,

ign) immediately before accented
e.g. café = caf¢e or für = f¢ur.

eft and) for right.

left and > for right.

full stop. Denomination changes
# sign, i.e., 9.25 a.m. = 9#25 a.m.

Equals	$= must be used to differentiate from the letter sign.
Interlining	$DOUBLE before and after a piece of text gives automatic double line spacing. It is automatically cancelled at the end of a file. This is used in young children's books (primary age) and in articles for braille learners.
Italics	Underscore _ before each word for 1, 2 or 3 words. For 4 of more words type two underscores __ before the first word and one underscore _ before the last, e.g., __For 4 or more _words. TYPED UNSPACED FROM WORD.
Letter sign	=, e.g., from =A to =Z; postal code =CV4 7AL. TYPED UNSPACED FROM LETTER. Used to distinguish letters not forming words, but not required if immediately after a digit.
New Line	$L
New Page	$PG
Poetry	Begin with $P $PY. Start each subsequent verse with $P. At the end of each line, except the last line of the poem, type $PS WITHOUT ANY PRECEDING SPACE. Applies to songs and hymns.
Pound sterling	Use L for pound (£) sign, e.g., £500 is L500.
Quotes	" for both left and right. Inner quotes (quotes within a quote) type: Left quote: $' leaving a SPACE before commencing word. Right quote: $'R UNSPACED FROM preceding word. e.g. $' The rules for inner quote$'R. An asterisk in the left hand margin of the line printer listing denotes lines where quotes are used. Check that quotes are correctly typed. If the quotes are odd in number, assuming you have missed one out, this will be stated at the end of the line printer listing.

Roman numerals	= sign, e.g., =I =VIII
Skip line(s)	$SLnn nn represents number of lines to be skipped, i.e., $SL01 = skip 1 line. $SL15 = skip 15 lines. The zero must be included if the number is less than 10.
Tabs	$#n where n is the number of the tab. There are 9 tabs numbered 1 to 9 which are initially set every two columns starting with column 3. e.g. $#2 would cause the following text to start in column 5. If it is necessary to change the position of a tab, $STBnLmm should be used, where n is the number of the tab and mm is the new column which the tab is to refer to. e.g. after $STB2L04, any occurrence of $#2 would cause the following text to start in column 4.
Uncontracted braille	$G before and after the appropriate words. Foreign words are an example where uncontracted words are used, e.g., $G Aberich glaube, er ist gut $G. If $G are odd in number, assuming you have missed one out, this will be stated at the end of the line printer listing.

QUICK REFERENCE:

Two most common commands:

Heading	$HDS ________ $HDE
Paragraph	$P

Alphabetical

Accent:	¢
Brackets	()
Brackets (square)	<>
Equals	$=
Italics	Underscore (see full list)
Letter sign	=

New Line	$L
New Page	$PG
Poetry	(see full list)
Pound sterling	L
Quotes	"
Quotes, inner	$'space/no space$'R
Roman numerals	=
Skip Line	$SLnn
Tabs	$#n (to change position) $STBnLmm
Uncontracted braille	$G ______ $G

FORMAT

Format on the whole remains as the printed copy, for the sake of clarity sometimes modifications are necessary. Underlining and ornamental type are disregarded. Capital letters are also rarely used in English braille.

Subjects within a document need some form of division, i.e., address at the top of a letter, ex. 2; subjects under side-headings, ex. 3. Imagine print with no block capitals and no underlining to catch the eye for side headings, etc. Likewise, in braille, the finger needs something clear to pick-up if reference documents are to be effective. General paragraphing in many documents is clear enough but, when side headings incorporate more than one paragraph a blank line should be left between subjects ($SL01). Another form of division, when more than one article appears in a document, is the centring of three spaced asterisks, i.e., $HDS * * * $HDE.

Some examples of format are given below:

EXAMPLE 1 - General:

LADDIE TO THE RESCUE

This is a true story. There are two boys in the story - David and Christopher - but the hero of the

Text input:

```
$HDS LADDIE TO THE RESCUE $HDE $P THIS IS A TRUE
STORY.  THERE ARE TWO BOYS IN THE STORY - DAVID
AND CHRISTOPHER - BUT THE HERO OF THE
```

EXAMPLE 2 - Letters: (for quick reference the name is shown before the address).

1 Ash Road,
Birmingham, B3 4TS

16th Sept., 1975.

Dear Mr. Smith,

Thank you for your letter and invite. I am pleased to accept and look forward to meeting you.

I remain,

Yours sincerely,

Joe Brown.

Text input:

```
MR JOE BROWN, $L 1 ASH ROAD, $L BIRMINGHAM =B3 4TS.
$L 16TH SEPT., 1975. $SL01 DEAR MR. SMITH, $P THANK
YOU FOR YOUR LETTER AND INVITE. I AM PLEASED TO
ACCEPT AND LOOK FORWARD TO MEETING YOU. $P I REMAIN,
YOURS SINCERELY.
```

EXAMPLE 3 - Minutes: (for clarity in braille a blank line is left between sections)

Committee for the Physically Handicapped
J. Briggs Chairman

Minutes for the meeting of the Committee for the Physically Handicapped. Held Tues 12 July, 1975. 7.30 p.m.

Present:

Mr. J. Briggs - Chairman
Mr. L. Smith - Vice Chairman
Mr. F. Brown - Social Services Dept.

29/75 MINUTES OF THE MEETING HELD 21 MAY 75

The minutes above corrected and approved.

MATTERS ARISING:

Car Parking for disabled. Mr. Brown reported

Text input:

```
$P COMMITTEE FOR THE PHYSICALLY HANDICAPPED. J. BRIGGS:
CHAIRMAN $P MINUTES FOR THE MEETING OF THE COMMITTEE
FOR THE PHYSICALLY HANDICAPPED. HELD TUES 12 JULY, 1975.
7 30 P.M. $P PRESENT $P MR. J. BRIGGS - CHAIRMAN $P
MR. L. SMITH - VICE CHAIRMAN $P MR. F. BROWN - SOCIAL
SERVICES DEPT. $SL01 $P 29/75 MINUTES OF THE MEETING
HELD 21 MAY 75. $P THE MINUTES ABOVE CORRECTED AND
APPROVED. $SL01 $P MATTERS ARISING $P CAR PARKING FOR
DISABLED. MR. BROWN REPORTED
```

EXAMPLE 4 - Statistics:

In braille it is impracticable to show statistics in tabular form. The headings are set out as a paragraph, each heading separated by a semi colon. The leader to the paragraph reading: "In the following table the column headings are:"

Age	Nature of Accident	Loss of Work	Compensation
39	Spine fracture	1 year	£1,500
56	Broken foot	18 weeks	Nil

Text input:

$P IN THE FOLLOWING TABLE THE FOUR COLUMN HEADINGS ARE: AGE; NATURE OF ACCIDENT; LOSS OF WORK; COMPENSATION. $P 39; SPINE FRACTURE; 1 YEAR; L1,500. $P 56; BROKEN FOOT; 18 WEEKS; NIL.

EXAMPLE 5 - Tables

BUS TIMETABLE

Monday-Friday				
Hanley (Bus station)	0630	0655	0715	0830
Smallthorne	0639	0704	0724	0804

Text input:

$HDS BUS TIMETABLE $HDE $P MONDAY-FRIDAY $P HANLEY (BUS STATION) 0630 ; 0655; 0715; 0830. $P SMALLTHORNE 0639; 0704; 0724; 0804.

EXAMPLE 6 - Poetry

The Dewdrops quiver on the cobweb tents,
Birds leave their love and sit in meek suspence.

A disk of fire aeons old cuts through
The rocks of earth and rolls up into view.

Text input:

$P $PY THE DEWDROPS QUIVER ON THE COBWEB TENTS,$PS BIRDS LEAVE THEIR LOVE AND SIT IN MEET SUSPENSE.$PS $P A DISK OF FIRE AEONS OLD CUTS THROUGH$PS THE ROCKS OF EARTH AND ROLLS UP INTO VIEW.

c.1
HV1703 D460
DESIGN AND EVALUATION OF A SYSTEM FOR THE PRODUCTION OF SHORT DOCUMENTS IN CONTRACTED BRAILLE.
(1975)
Date Due

c.1
HV1703
D460
DESIGN AND EVALUATION OF A SYSTEM FOR THE PRODUCTION OF SHORT DOCUMENTS IN CONTRACTED BRAILLE.
(1975)

DATE	ISSUED TO
	Reference Copy

AMERICAN FOUNDATION FOR THE BLIND
15 WEST 16th STREET
NEW YORK, N. Y. 10011

Printed in U.S.A.